

# **JasperServer User's Guide**

**Version 3.5**

© 2009 JasperSoft Corporation. All rights reserved. Printed in the U.S.A. JasperSoft, the JasperSoft logo, JasperAnalysis, JasperServer, JasperETL, JasperReports, JasperStudio, iReport, and Jasper4 products are trademarks and/or registered trademarks of JasperSoft Corporation in the United States and in jurisdictions throughout the world. All other company and product names are or may be trade names or trademarks of their respective owners.

---

**Table of Contents**

1 Introduction..... 1

    1.1 What is JasperServer?..... 1

    1.2 Who is this guide for?..... 1

    1.3 Related documents..... 1

2 Terms and concepts..... 1

    2.1 JasperServer Report ..... 1

    2.2 JRXML..... 1

    2.3 Resource..... 2

    2.4 Data source..... 2

    2.5 Query..... 2

    2.6 Image..... 2

    2.7 Input control..... 2

    2.8 Data type..... 2

    2.9 Analysis view..... 3

    2.10 OLAP client connection..... 3

    2.11 Mondrian connection..... 3

    2.12 XML/A connection ..... 3

    2.13 Mondrian XML/A source..... 3

    2.14 Analysis schema..... 3

    2.15 MDX query..... 3

3 Accessing Reports and Analytics..... 4

    3.1 Running a report..... 4

        3.1.1 Running a report from the repository browser..... 4

        3.1.2 Running a report through direct URL..... 6

    3.2 Using Report Scheduler..... 8

        3.2.1 Report browser..... 8

        3.2.2 Report jobs list..... 8

        3.2.3 Report job details..... 9

        3.2.4 Quick background execution..... 13

        3.2.5 Built-in report parameters..... 13

4 JasperServer Administration..... 14

    4.1 Creating a folder..... 14

    4.2 Creating a simple report..... 15

        4.2.1 Naming the new report..... 15

        4.2.2 Add the JRXML file, logo, and data source..... 16

    4.3 Creating a complex report..... 22

        4.3.1 Naming the new report..... 22

        4.3.2 Add the JRXML file, logo, and data source..... 22

        4.3.3 Adding input controls..... 23

        4.3.4 Adding external resources..... 28

        4.3.5 Adding a datasource..... 29

        4.3.6 More on data types..... 30

    4.4 Creating reports using the iReport plugin..... 30

        4.4.1 Plugin features..... 30

        4.4.2 Making connection to JasperServer..... 32

        4.4.3 Using the plugin..... 34

        4.4.4 Folders..... 34

        4.4.5 Importing JDBC connections into iReport..... 35

4.4.6	Creating a JasperServer Report from iReport.....	36
4.4.7	Edit a JRXML.....	37
4.4.8	Running a JasperServer Report from iReport.....	37
4.4.9	Adding an image to a report.....	38
4.4.10	Adding an subreport to a report.....	40
4.4.11	Resources, input controls, data types, lists of values and data sources.....	41
4.4.12	Repository resource chooser.....	41
4.5	Adding resources directly to the repository.....	42
4.6	Move/Copy resources in the repository.....	43
4.6.1	Copying resources.....	43
4.6.2	Moving a folder.....	44
4.7	Security Maintenance.....	45
4.7.1	Overview.....	45
4.7.1.1	Authentication.....	45
4.7.1.2	Authorization.....	46
4.7.2	Creating a role.....	46
4.7.3	Creating a user.....	47
4.7.3.1	Changing role assignments for existing users.....	49
4.7.4	Configuring Password Options.....	50
4.7.4.1	Changing Password Encryption Settings.....	50
4.7.4.2	Allowing Users to Change Their Passwords.....	51
4.7.4.3	Enabling Password Expiration.....	52
4.7.5	Setting Permissions.....	52
5	Utilities.....	52
5.1	Import/Export.....	52
5.1.1	Import Resources.....	52
5.1.2	Export Resources.....	53
5.2	Message viewer.....	54
5.2.1	Messages list.....	54
5.2.2	Message details.....	55
5.3	iReport Plugin installation and configuration.....	55
5.3.1	Requirements.....	55
5.3.2	Plugin installation.....	56
5.3.3	Configuring the plugin.....	57
5.3.4	Troubleshooting.....	59
6	Advanced configuration.....	59
6.1	Custom Data Sources.....	59
6.1.1	Background on data sources in JasperServer and JasperReports.....	59
6.1.1.1	Query executers.....	60
6.1.2	Custom Data Source Examples.....	60
6.1.2.1	Custom Bean Data Source.....	61
6.1.2.2	Webscraper Custom Data Source.....	61
6.1.3	Creating a Custom Data Source.....	62
6.1.3.1	Files used by a custom data source implementation.....	62
6.1.3.2	Implementing the ReportDataSourceService Interface.....	62
6.1.3.3	Defining Custom Data Source Properties.....	62
6.1.3.4	Implementing Optional Interfaces.....	62
6.1.3.5	Creating the Message Catalog.....	63
6.1.3.6	Defining the Custom Data Source in Spring.....	63

6.1.3.7	Configuring the Message Catalog.....	64
6.1.4	Installing a Custom Data Source.....	64
6.1.5	Using a Custom Data Source.....	65
6.2	Report output channels.....	65
6.2.1	Excel output.....	65
6.2.2	CSV output.....	66
6.3	Configuring Login Page.....	66
7	Integrating JasperServer and Liferay Portal.....	66
7.1	Changing Liferay's Port Numbers .....	66
7.2	Configuring JasperServer to Accept Web Services Calls .....	67
7.3	Configuring Liferay to Access JasperServer .....	67
7.4	Testing Liferay .....	68
7.5	Deploying the JasperServer Portlet WAR File .....	68
7.6	Configuring a Default Report to Display.....	69
7.7	Testing the JasperServer Portlet.....	70
7.8	JasperServer Portlet Configuration Options.....	70
7.8.1	Portlet Parameters.....	70
7.8.1.1	image_servlet_ssl_enabled.....	70
7.8.1.2	portal_server.....	70
7.8.1.3	show_logo.....	70
7.8.1.4	show_return_to_report_list_icon.....	70
7.8.1.5	show_return_to_parameter_icon.....	71
7.8.1.6	company_logo.....	71
7.8.1.7	company_logo_hyper_link.....	71
7.8.1.8	report_directory.....	71
7.8.1.9	number_of_reports_per_page.....	71
7.8.2	Example Server and Browser Configuration.....	71
7.8.2.1	applicationContext-security.....	72
7.8.2.2	portlet.xml.....	72
7.8.2.3	Browser URL.....	72
7.9	Setting up JasperReports Hyperlinks for Use in a Portlet Environment.....	72

## 1 Introduction

### 1.1 What is JasperServer?

JasperServer is a framework that lets you build a Web application around JasperReports and use JasperAnalysis, an implementation of Online Analytical Processing or OLAP, to perform on-line analytical tasks. Using JasperReports, it lets you create, schedule and run reports, and manage resources, roles, and permissions. It also provides a sample Web-based user interface that lets you create and manage reports and other resources. Using JasperAnalysis, it lets you interact with analysis views to perform operations such as slice/dice, drill down/drill through, etc. It also lets you create new analysis views with OLAP schemas and various data sources.

### 1.2 Who is this guide for?

This guide is for users of the sample user interface provided with JasperServer.

With respect to JasperReports, in the current release, you must use iReport, or write JRXML code, to create the JRXML file that is the basis of the report. The sample JasperReports UI only assembles the JRXML file and other resources into a report that you can run. This UI cannot create a JRXML file.

**Note:** *To use this user interface, you must be very familiar with the JasperReport (.jrxml) files that are the basis of the JasperServer report. It is assumed that you understand JRXML and you are very familiar with all the resources that a JRXML file refers to.*

With respect to JasperAnalysis, in the current release, you must supply OLAP schemas in XML format, corresponding data sources as JDBC or JNDI, and MDX queries. The sample JasperAnalysis UI only assembles the schemas, data source and MDX queries into analysis views that you can interact. This UI cannot create schemas or MDX queries. For more information, refer to the JasperAnalysis User's Guide.

### 1.3 Related documents

- JasperServer Installation Guide
- JasperAnalysis User's Guide
- JasperServer Developer's Guide
- JasperServer Web Services Client Install Guide
- JasperServer Database Scripts install Guide
- [The Definitive Guide to JasperReports](#)
- [iReport manual](#)

## 2 Terms and concepts

### 2.1 JasperServer Report

A *JasperServer report* is what you actually run to generate output. A report consists of a set of [resources](#). (See the [Resource](#) section.)

A JasperServer report may contain resources in either the file system or the repository.

### 2.2 JRXML

*JRXML* is JasperReport's XML-based report definition language. Every JasperServer report must contain one main JRXML file and one for each subreport it might contain.

## 2.3 Resource

A *resource* is one of the following elements of a report:

- a JRXML file (see the [JRXML](#) section)
- a data source (JNDI or JDBC)
- a query
- fonts
- images
- data types (see the [Data type](#) section)
- JAR files for JasperReports scriptlets
- resource bundles for localization and internationalization
- subreports

Every [JasperServer report](#) must contain a [JRXML](#) file and a [data source](#). The other [resources](#) may or may not be required, depending on the contents of the JRXML file. Therefore, to create a report, you must know the resources required by the JRXML file.

You can retrieve resources from the file system or the repository.

## 2.4 Data source

The three supported data source types are JDBC, JNDI and Bean.

For reports with MDX queries in their main JRXMLs, [OLAP client connections](#) are used as data sources. Such reports use the Mondrian and XML/A query executors included in JasperReports when launched.

## 2.5 Query

Instead of the query stored inside the JRXML file, a report can make use of a reusable query from the repository.

## 2.6 Image

You can point to an image in one of the following three ways:

- using a URL
- using the classpath
- using the "repository" syntax (i.e., repo:/image/logo.jpg)

## 2.7 Input control

*Input controls* are values that can be mapped to JasperReports parameters.

## 2.8 Data type

In JasperServer, a *data type* is a set of constraints on the value of an [input control](#). It contains more information

---

(for example, maximum and minimum values) than a data type in most programming languages.

## **2.9 Analysis view**

An *analysis view* consists of an [OLAP client connection](#) and a [MDX query](#) for providing an entry point to OLAP operations, such as slice/dice, drill down and drill through.

## **2.10 OLAP client connection**

An *olap client connection* contains the connection definitions for retrieving an [analysis view](#). The two types of OLAP client connection are [Mondrian connection](#) and [XML/A connection](#).

## **2.11 Mondrian connection**

A *Mondrian connection* is a type of [OLAP client connection](#), which consists of an [OLAP schema](#) and a [data source](#) used to access an [analysis view](#).

## **2.12 XML/A connection**

An *XML/A connection* is a type of [OLAP client connection](#), which consists of SOAP definitions used to access an analysis view on a remote XML/A Provider.

## **2.13 Mondrian XML/A source**

A *Mondrian XML/A source* is a server-side XML/A source definition for defining how XML/A connections will be processed from remote clients to access an [analysis view](#) via a [Mondrian connection](#) on this server.

## **2.14 Analysis schema**

An *analysis schema* is a file [resource](#) for defining and documenting the structure of OLAP cubes, dimensions and measures in XML format.

## **2.15 MDX query**

*MDX query* is a language for querying multidimensional objects, such as cubes, and return cube data for analytical processing.

### 3 Accessing Reports and Analytics

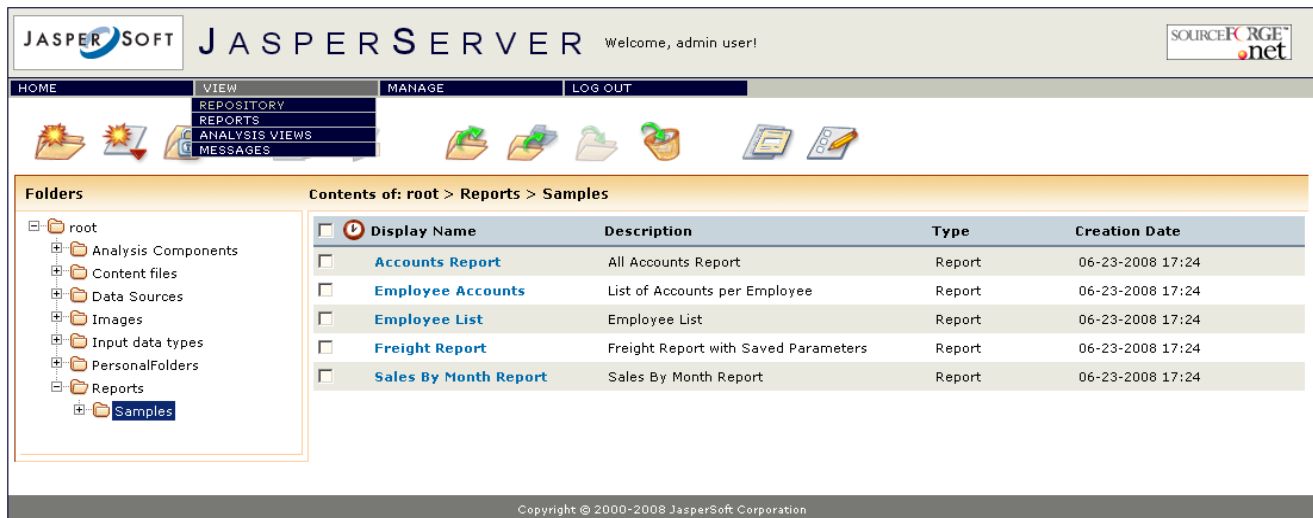
The user interface features four menu buttons:

Menu name	Function
Home	Describes JasperServer.
View	Shows metadata for, and lets you run, reports in the repository. All reports for which you have read permission appear regardless of which folder they're in.
Manage	Contains the Repository, Roles, and Users submenus. It appears only if you have administration privilege.
Logout	Logs you out of JasperServer.

#### 3.1 Running a report

First, you'll run the sample SalesByMonth report. This report contains four input controls: a text input control, a Boolean checkbox, a list input control, and a date input control.

##### 3.1.1 Running a report from the repository browser



- Mouse on the VIEW menu.
- Click the REPOSITORY menu item.
- Click the samples folder
- Click the SalesByMonth sample. Then click on the Report Options button in the left upper corner. A list of controls appears.
- For the Text Input Control, enter 999.
- Check the Checkbox Input Control.
- In the List Input Control dropdown list, select Yet Another Item.
- In the Date Input Control, click the calendar icon (on the right).
- Choose a date.



**Report Options: Sales By Month Report**

Text Input Control

Checkbox Input Control

\* List Input Control

DateInput\_label

QueryInput\_label

- Click OK and wait for the report to run. Notice the values in the report title. These are derived from the values you gave for the input controls. Navigate through the report using the left- and right-arrow controls at the top of the report.

JASPER SOFT JASPER SERVER Welcome, tomcat user! SOURCEFORGE net


HOME VIEW MANAGE LOG OUT

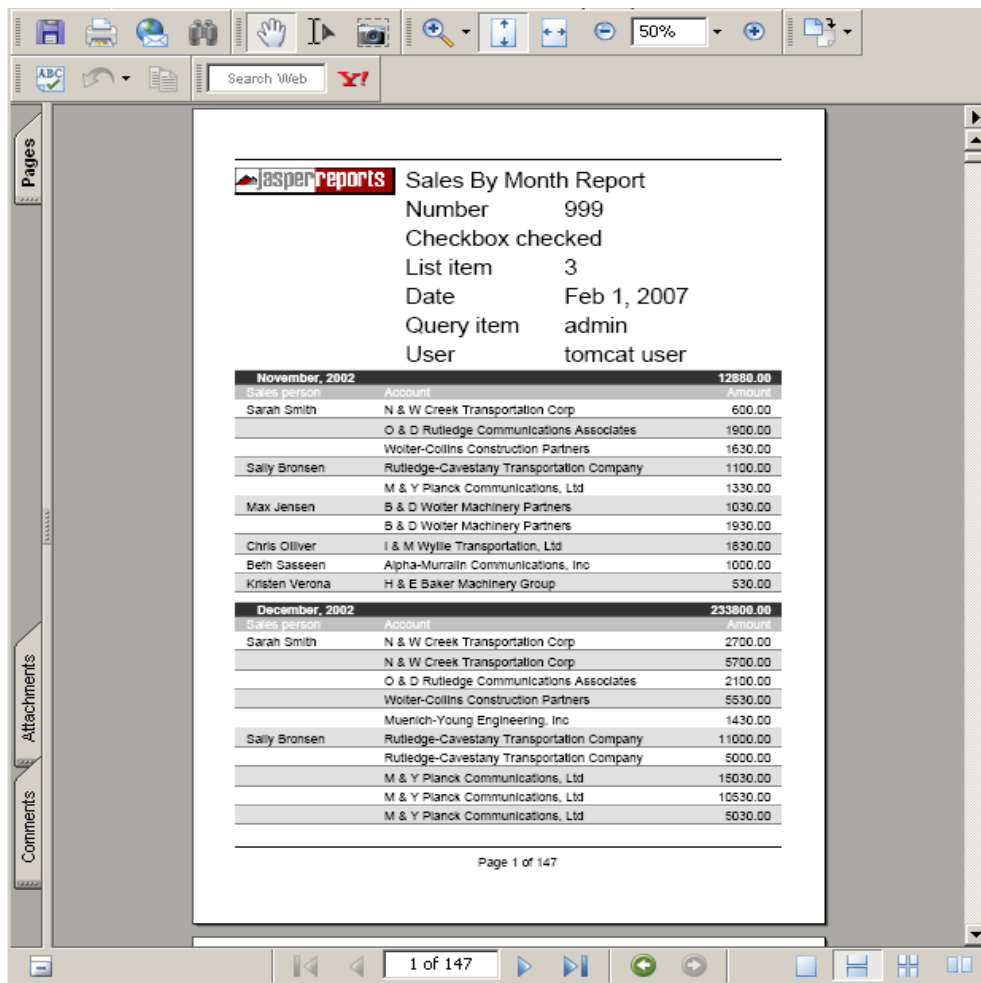
Page 1 of 152

**jasperreports** Sales By Month Report

Number 999  
 Checkbox checked  
 List item 3  
 Date Feb 1, 2007  
 Query item admin  
 User tomcat user

November, 2002		12880.00
Sales person	Account	Amount
Sarah Smith	N & W Creek Transportation Corp	600.00
	O & D Rutledge Communications Associates	1900.00
	Wolter-Collins Construction Partners	1630.00
Sally Bronsen	Rutledge-Cavestany Transportation Company	1100.00
	M & Y Planck Communications, Ltd	1330.00

- Click the PDF icon. The report appears in a separate browser in PDF format.
- Click the Close icon (  ) to close the report.



### 3.1.2 Running a report through direct URL

Reports stored in JasperServer can be executed via direct URLs that can be used or embedded into external applications. All the information required to execute a JasperServer report can be passed as URL parameters.

A report execution URL has the following syntax:

```
<JS deployment context>/flow.html?_flowId=viewReportFlow<report execution parameters>
```

The first part is the base URL of the JasperServer deployment, for instance

```
http://server:8080/jasperserver
```

if JasperServer is deployed as `jasperserver` on the `server` machine on port 8080.

The report execution parameters can be either reserved parameters used by JasperServer to determine general attributes of the report execution, or arbitrary parameters that correspond to the report input controls/parameters. The parameters are specified as standard HTTP GET parameters, i.e. in the form of `name=value` and separated by `&` characters.

The following general parameters are recognized by JasperServer:

- `reportUnit` is the URI of the report unit resource in the JasperServer repository.
- `output` (optional) specifies the desired report output format. If it is not present or `html` is used as value for the parameter, the default JasperServer report viewer is used as output. Other values for this parameter are given by the report exporters configured in JasperServer. By default,

JasperServer recognizes the following output types: `pdf` for PDF, `xls` for Excel, `rtf` for RTF, `csv` for CSV and `swf` for the JasperServer Flash report viewer.

- `reportLocale` is used to pass the locale in which the report is to be executed. A locale is passed as code consisting of a lower-case two letter ISO-639 language code, followed by optional upper-case two letter ISO-3166 country code and a locale variant, separated by underscore (the format is identical to the standard Java locale programmatic names).
- `j_username` and `j_password` can be used to pass the credentials to be used to authenticate a user with JasperServer. The username should correspond to a valid JasperServer user, and password should be the user password (in clear text).

If such credential parameters are not present, and no authenticated JasperServer session exists, and JasperServer is not configured to use automatic authentication mechanisms (such as Single sign-on), the user accessing a report execution URL will be first required to provide a username and a password on the login screen, and then redirected to the actual report execution screen. The two authentication parameters can be used to skip the login screen and have the user directly presented with the report execution screen.

Note that these two parameters are not specific to report execution URLs, they can be used for any URLs that point to a JasperServer web page.

Apart from these standard parameters, report execution URLs can contain parameters that provide values for the report input controls/parameters. The URL parameter names must match the name of the corresponding report input control. The URL parameter values are specified depending on the input control type:

- For simple single value input controls, the desired value is used as URL parameter value:
  - If the type of the input control is text, the URL parameter value is directly used as input control value.
  - If the type of the input control is numeric, the URL parameter value is the numerical value formatted according to standard rules, using a dot as decimal separator.
  - If the type of the input control is date or date/time, the URL parameter value is the date/time value formatted according to the `yyyyMMddHHmmss` format (e.g. `20090621054500` for June 21st 2009, 05:45). The formatted value is interpreted according to JasperServer's default timezone.
- For boolean/checkbox input controls, the URL parameter value can be either `true` or `false`.
- For input controls backed by static list of values, the URL parameter value should be the key/value of the list entry.
- For query-backed input controls, the URL parameter value should correspond to the query key/value column.
- For multi-valued input controls, multiple occurrences of the same URL parameter can be used (e.g. `parameter=value1&parameter=value2&parameter=value3`).  
A special marker URL parameter can be used for multi-valued input controls to specify that an empty list should be used as input control value; the marker URL parameter uses as name the name of the input control prefixed by an underscore and doesn't require any value. This is useful when a multi-valued input control has a non-empty list as default value, and the user wants to override the default value with an empty list.

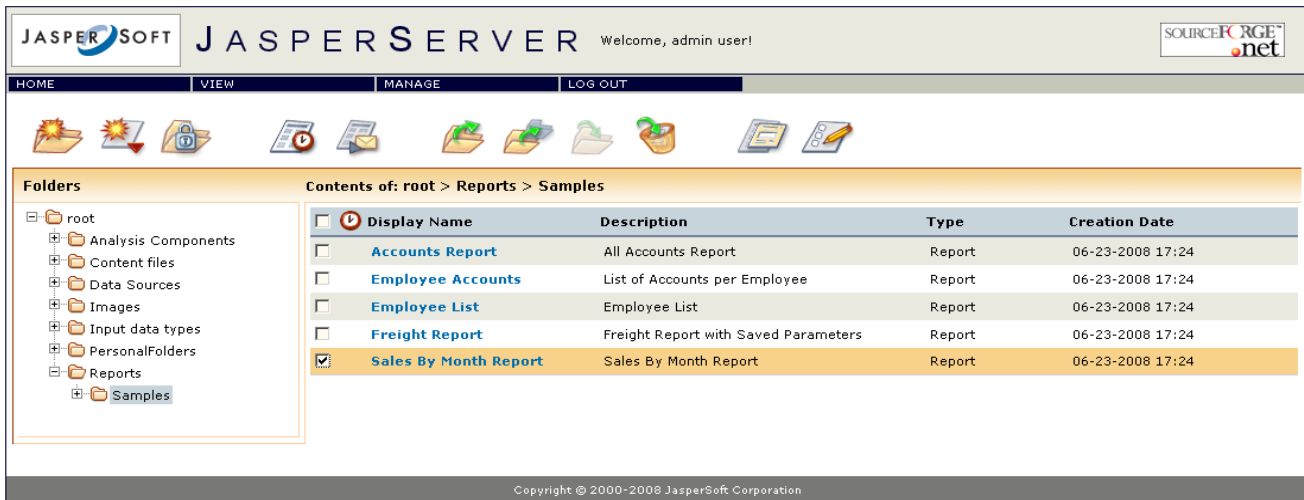
If the report parameter values included in a report execution URL are not valid (i.e. if a required parameter is missing or a parameter value is not valid), the report input controls are displayed to the user so that they can be corrected.

### 3.2 Using Report Scheduler

The report scheduler runs report execution jobs in the background and saves the execution result in the repository. Users can define such jobs and schedule them to run at specific moments in the future and find the resulting reports in the repository or receive the results by email.

#### 3.2.1 Report browser

In the [report browser](#), there are two icons that provide access to the report scheduling functionality.



The two icons have the following purposes:

- The "Schedule report" (🕒) icon open the [list of jobs](#) defined for the report.
- The "Run in background" (📧) provides a quick way to run a report in background (asynchronously), i.e. to schedule a report execution job that would get executed immediately. More about this [here](#).

#### 3.2.2 Report jobs list

When the "Schedule report" icon is clicked, the [list of report jobs](#) opens.



Each regular user only sees his own jobs, while administrative users will see all the report jobs.

The following information is displayed in the jobs list:

1. the job ID, accompanied by a link that opens the job for editing.
2. the job label, which is a short description provided while creating the job
3. the job owner, i.e. the user that created the job
4. the job state, which is one of:
  1. Normal: the job is scheduled
  2. Running: the job is currently running
  3. Complete: the job has completed all its executions.
  4. Error: the scheduler encountered an error while scheduling or triggering the job. This does not include the cases when the job is successful triggered, but some error occurs during execution.
5. the last time the job was fired
6. the next time the job will get fired

Several controls are placed on the report jobs list:

1. The "Back" button is used to return to the report browser screen.
2. The "Schedule Job" button is used to create a new job for the report. Clicking it opens the [job details](#) screen.
3. The "Run Now" has the same purpose as the "Run in background" icon on the report browser screen (more details [here](#)).
4. The "Refresh" button is used to refetch from the server an updated jobs list.
5. The "Remove" button is used to delete the selected jobs. Jobs are selected via the checkboxes placed on all rows; the checkbox placed on the list header can be used to select/unselect all the jobs.

When a jobs gets deleted, it will be unscheduled but if the job is currently running the execution will not be stopped. Completed report jobs, i.e. jobs that will not be fired again, will be automatically deleted from the scheduler.

### 3.2.3 Report job details

The report job details consist of information required to schedule a report execution:

- schedule information: when should the report be executed
- report parameters
- output: where should the resulting report be sent

#### Job | Schedule | Parameters | Output

Report	/reports/samples/SalesByMonth
* Label	Monthly report
Description	Outputs monthly accounts

The job details screens are used both for scheduling a new job and for viewing/updating an existing one.

The [first screen](#) contains general job description attributes:

- The job label is a short description that will get displayed in the jobs list.
- The job description (optional) is a long job description.

The [second screen](#) is used to define job scheduling information. This information consists of job start and end times and from optional recurrence information. The user can set the following scheduling attributes:

- A time zone to use for the scheduling dates. By default, the scheduling dates and times are interpreted in the server time zone, but the user has the option to specify the dates in another time zone and the scheduler will take this into account when firing the job.
- A start date to indicate when the job should become effective. The user has the option of instructing the job to start immediately or at a specific date and time in the future. The start date has slightly different meaning depending on the recurrence type. If no recurrence is defined, the start date will be the date the job is going to be executed at.

**Job | Schedule | Parameters | Output**

Time Zone

Start  Immediately  
 On

No Recurrence  Simple Recurrence  Calendar Recurrence

Apart from this attributes, the user can optionally specify a recurrence scheme for the job. There are two types of recurrence mechanisms that can be used: simple recurrence and calendar recurrence.

The simple recurrence mechanism allows the user to schedule the job to recur at fixed time intervals.

No Recurrence  Simple Recurrence  Calendar Recurrence

Occur  Indefinitely  
 Until   
  times

\* Every


The user can set the following attributes:

- Maximum number of occurrences is optional and specifies the maximum number a job can occur. The job would occur until either the occurrence count reaches this value or the end date (if defined) is reached. The user can also choose not to specify a maximum number of occurrences and let the job recur indefinitely or until the end date.
- The fixed recurrence interval, composed of a numerical value and a unit of measure. The user can specify the interval in minutes, hours, days or weeks.

In case of simple recurrence, the job start date will be the date the job is going to be first fire at and the next occurrences will be computed based on the start date and the recurrence interval.

The [calendar recurrence](#) mechanism allows users to define jobs that recur at specific calendar moments.

No Recurrence  
  Simple Recurrence  
  Calendar Recurrence

End date  

\* Minute(s)

\* Hour(s)

Days  Every Day

Week Days

Mon Tue Wed Thu Fri Sat Sun

Month Days

Months  All

Jan  Feb  Mar  Apr  May  Jun

Jul  Aug  Sep  Oct  Nov  Dec

The End date field specifies the date limit until the job will be performed. If it is not specified, the job will run indefinitely.

The user needs to specify the minutes, hours, days and months the job should occur at:

- One can specify one or more minutes the job should be executed at. The accepted values are in the 0 to 59 range. In the most common scenarios, only one value would be set for the minutes. If the report needs to be executed multiple times per hour, the minutes can be specified using the following syntax:
  - X,Y,Z,... – the job will be executed at minutes X, Y, Z, ..
  - X-Y – the job will be executed every minute from X to Y
  - \* - the job will be executed every minute
- The hour(s) field is similar to the minutes, with the difference that the accepted values are in the 0 to 23 range.
- To select the days the job should run on, the user has three options:
  - Set the job to run every day.
  - Set the job to run on specific week days.
  - Set the job to run on specific month days. The month days field is similar to the minutes field, with accepted values in the 1 to 31 range.
- The months the job should be executed on are selected individually. A checkbox to select/deselect all months is available for convenience.

If the user chooses calendar recurrence, the job will first be executed at the first moment after the start date that matches all the defined date and time values. The job would then recur at each matching date and time until the end date is reached or indefinitely if an end date is not set.

If the report has input controls, the job details will contain a set of input values which will be used when executing the report. A [separate screen](#), identical to the one used when directly executing the report, will allow

the user to input value for the report parameters.

[Job](#) | [Schedule](#) | [Parameters](#) | [Output](#)

Text Input Control

Checkbox Input Control

\* List Input Control

DateInput\_label

QueryInput\_label

Finally, the user has to define the job output attributes. Each time the job runs, the results are saved into the content repository. Optionally, the result can also be distributed by email to one or more recipients.

The following attributes can be set in the [job output screen](#):

[Job](#) | [Schedule](#) | [Parameters](#) | [Output](#)

Base output file name

Output description

Output formats  PDF  HTML  Excel  RTF  CSV

Locale

**Content Repository**

\* Folder

Sequential File Names

Timestamp Pattern

Overwrite Files

**Email Notification**

To

Subject

Message Text

Attach Files

Skip empty reports

- The base output filename is used to derive all output filenames by appending an optional timestamp and a format extension.
- The user needs to choose at least one output format. The available list includes PDF, HTML, Microsoft



Excel, RTF and CSV.

- The user can choose a locale that should be used when filling the report. If no specific locale is selected, the server will use the default locale.
- The content repository folder where all the resulting files should be saved has to be selected.
- The user can choose to include a timestamp in the file names used to save the job results. This is useful for recurring jobs when multiple results need to be kept.
- The user can instruct the scheduler whether overwriting existing content files is allowed. If the scheduler wants to save a report and another saved report having the same name already exists, it will either overwrite it or generate an error, depending on this flag.
- In addition to saving the result in the content repository, the job creator can choose to enable an email notification for the job. Each time the job would get executed, an email would be sent to one or more recipients that would optionally include the job results as attachments.

A user needs to specify the following attributes for the email notification:

- One or more recipient addresses, separated by comma.
- The message subject.
- The message body.
- Whether the job results should be attached to the message.
- Whether the mail should not be sent when the generated is empty.

As we mentioned before, the report job details screens are used for both creating a new job and for updating an existing job. The users should note that if the scheduling attributes of a job are updated, the scheduler will recreate the trigger that fires the job execution. This might have unexpected results if the job start date is in the past and should generally be avoided. If other job attributes are updated, they will take effect from the next job execution.

### **3.2.4 Quick background execution**

The "Run in background" report execution allows users to define a report job that would get fired immediately without manually entering all the job attributes. This functionality can be used to launch asynchronously long report executions and save the results in the repository (and optionally receive the results via email).

If the user wants to quickly launch a report execution in background, he will only be required to enter the report parameters if the report has input controls and to define the job output. The screens used for these tasks are the same as the ones used when scheduling a new report job.

### **3.2.5 Built-in report parameters**

When a report is executed by the report scheduler, a built-in parameter named "\_ScheduledTime" is used to pass to the report the date and time the report was scheduled at. The parameter needs to be declared in the JRXML in order to be used:

```
<parameter name="_ScheduledTime" class="java.util.Date"/>
```

The value can be used to filter the reported date or for displaying purposes.

## 4 JasperServer Administration

The Manage menu appears only if you have administrator privileges. It consists of the following submenus:

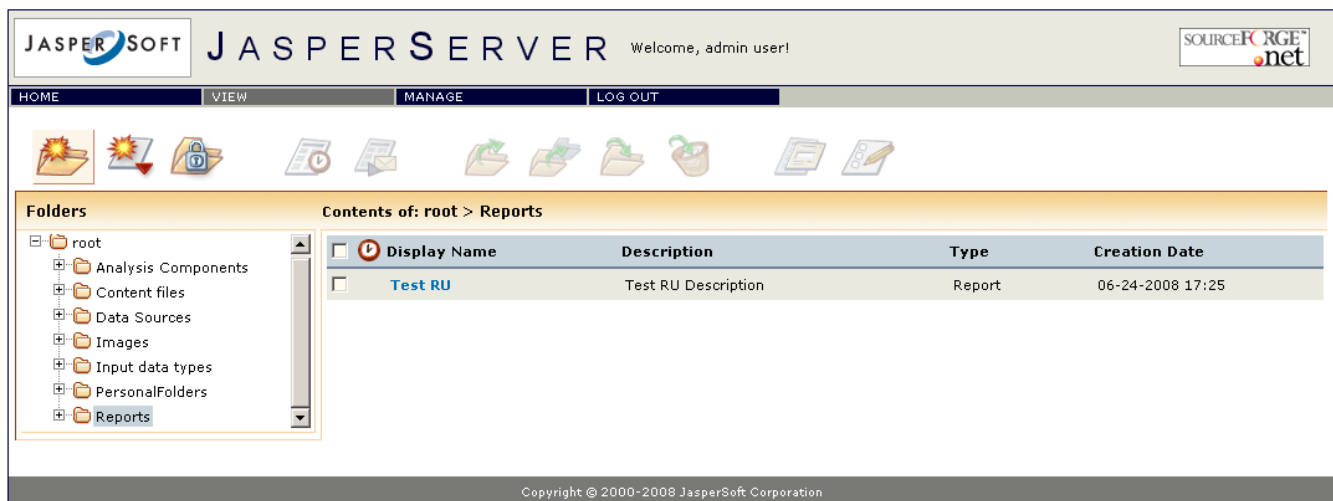
<b>Repository</b>	Displays the repository browser, which contains data sources, images, reports, and other user-created folders.
<b>Roles</b>	Lets you manage roles that you assign to users.
<b>Users</b>	Lets you see and administer users.
<b>Analysis Properties</b>	Displays current Mondrian properties
<b>Flush OLAP Cache</b>	Clears cache used by OLAP operations

### 4.1 Creating a folder

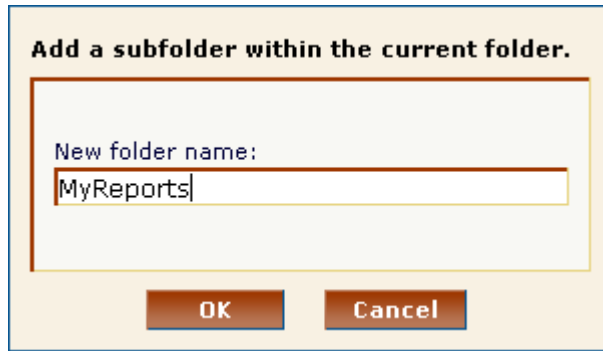
If you have administrator privileges, JasperServer lets you create a tree of folders in which to put your reports and resources. To do this:

1. Click the REPOSITORY submenu of the VIEW menu.
2. Choose or create the repository folder in which you want to create your new report. For this example, click reports. The path should display as follows:

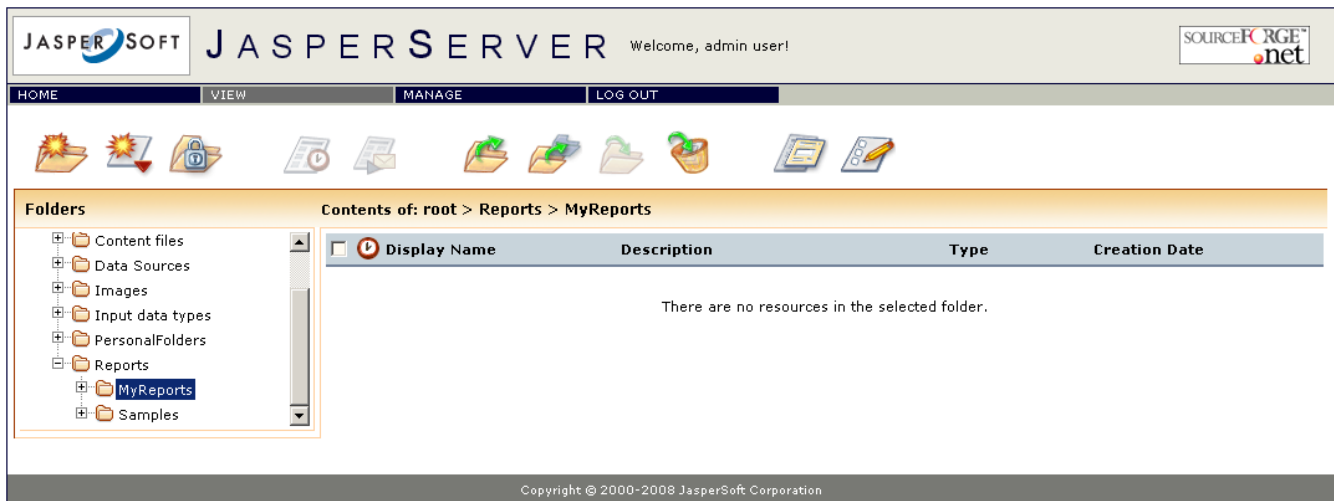
[root/Reports](#)



3. Click the *Add Folder* button in the toolbar. The Edit Folder screen appears.
4. Enter the name of the new folder you want to create. In this case, enter *MyReports*.



5. Click OK. The folder appears in the repository.



## 4.2 Creating a simple report

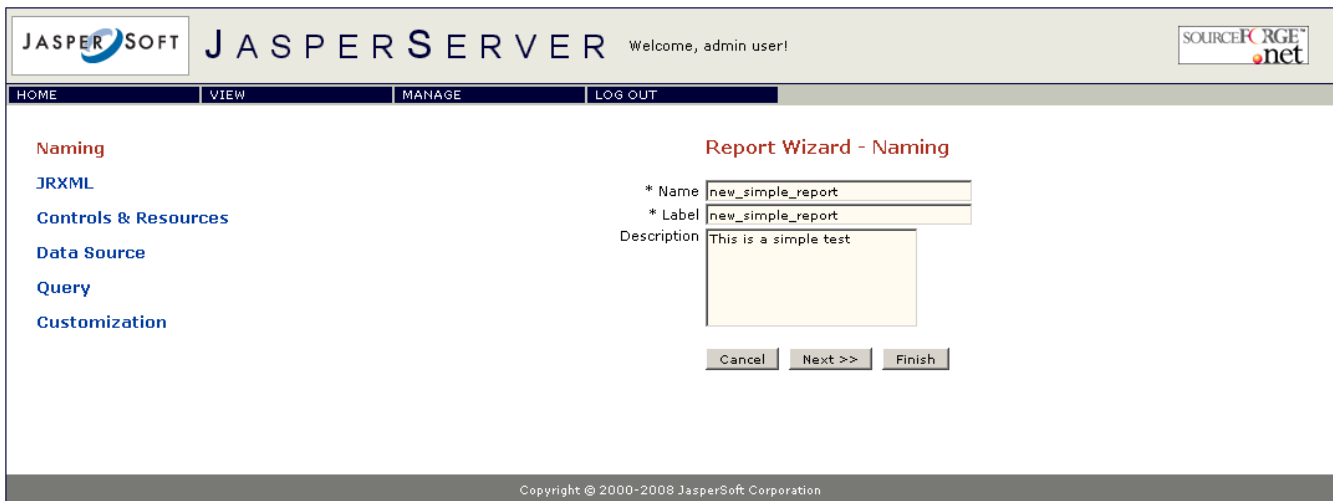
In this example, you'll create a report to run the *AllAccounts.jrxml* file. This is a simple report that takes no input parameters. To create the report, you must have access to the *samples/reports/AllAccounts.jrxml* and the *samples/image/logo.jpg*. Both of these resources can be found in the *samples* directory at the root of the JasperServer installation directory.

### 4.2.1 Naming the new report

1. In the REPOSITORY browser, select the *MyReports* folder that you have created.
2. Click the *Add Resource* button in the toolbar and select *JasperServer Report* in the pop-up menu that opens.



3. The Create Report Wizard appears.
4. For *Name*, enter *new\_simple\_report*.
5. For *Label*, enter *New Simple Report*.
6. For *Description*, enter *This is a simple test*.



7. Click Next.

#### 4.2.2 Add the JRXML file, logo, and data source

8. Now that you've named the report, you'll add the resources you need to run it. Under *Locate main JRXML File:*, click the *From file System* radio button.
9. Click the *Browse* button. Locate *samples/reports/AllAccounts.jrxml*. (You should find it in the *<JasperServer-install-dir>/samples/reports* directory). Select the *AllAccounts.jrxml* file.



10. Click *Next*.

11. The Resource List screen shows you need the *LogoLink* resource.



12. On the right side of the LogoLink line click *Add Now*.

13. Click the *From the Repository* radio button and select */images/JRLogo*.



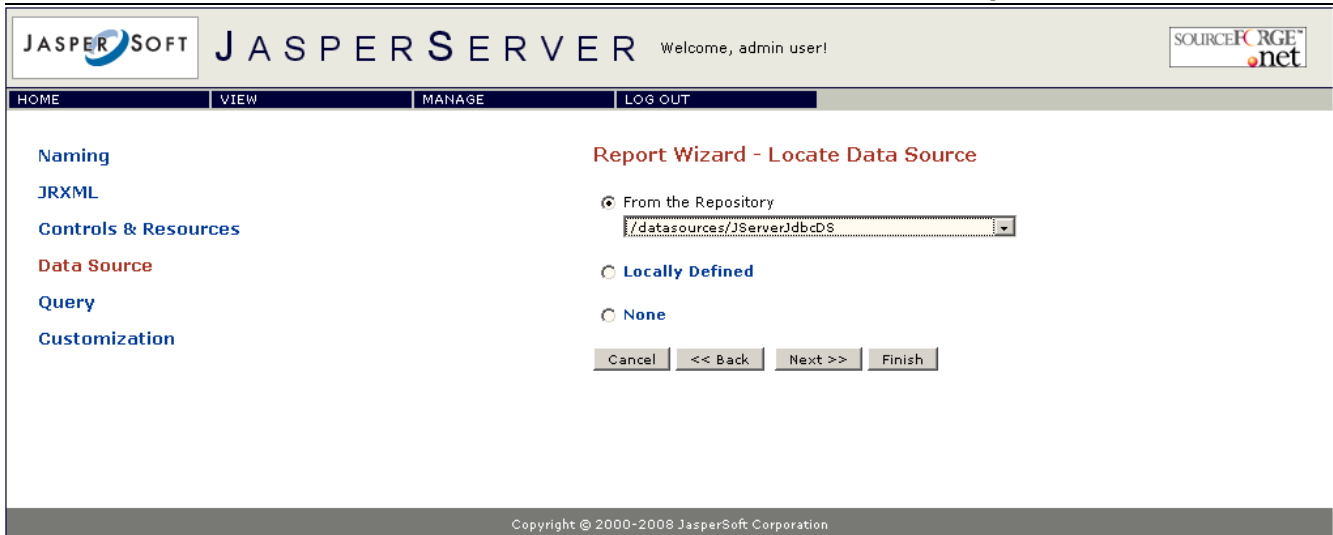
14. Click *Next*.

15. Accept the default naming of the image in the Repository. Click *Next*.



16. You should now back on the Resource List screen. Click *Next*.

17. On the Locate Data Source screen, click the *From the Repository* radio button.



18. Accept the default data source. Click *Next*.

19. Our JRXML file already has a query inside, but are going to override it by creating local query resource; pick *Locally Defined* in the *Locate Query* page. *None* is for using the JRXML query, if present.

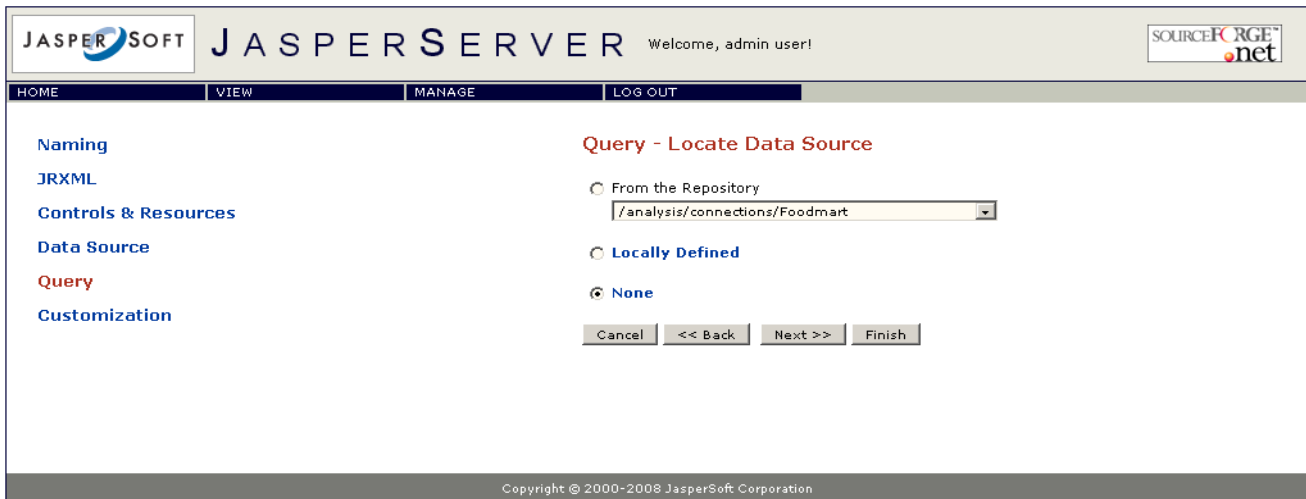


20. Name the local query *CanadaAccounts* and use the same for label.

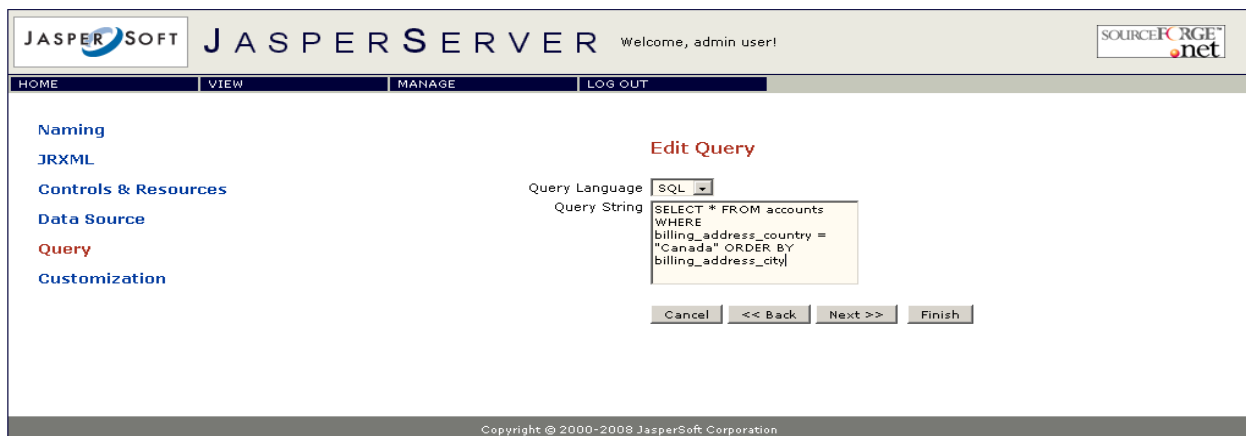
21. Click *Next*.



22. The query itself can point to a different data source. But we want to use the data source that was already picked for the report and thus pick *None*.

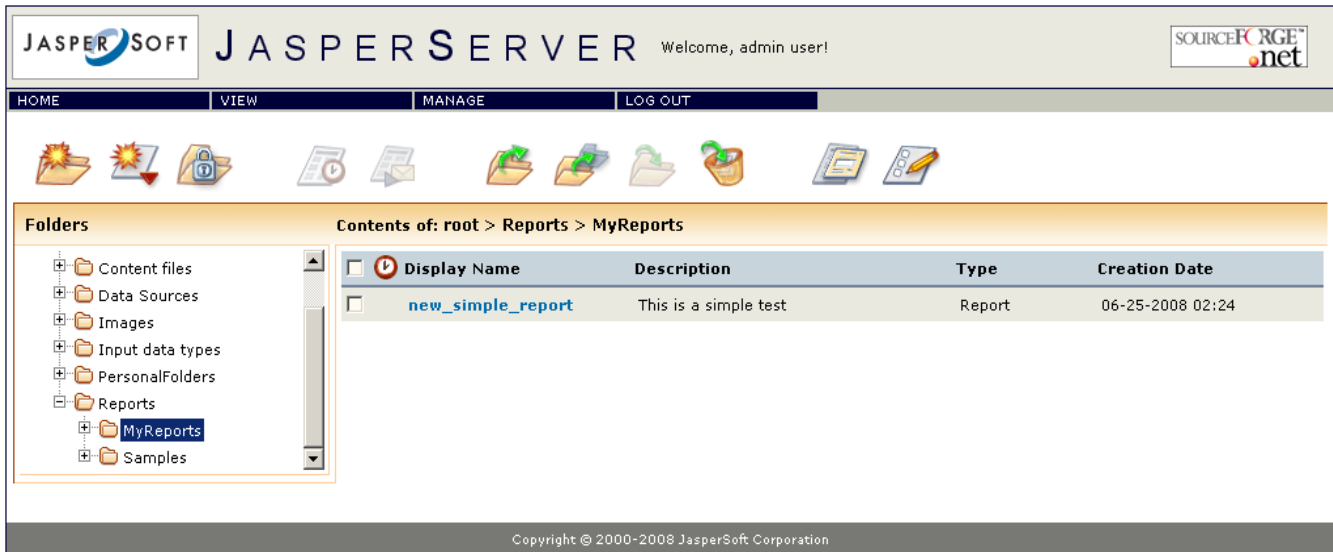


23. Use SQL as query language and the following query text: *SELECT \* FROM accounts WHERE billing\_address\_country = "Canada" ORDER BY billing\_address\_city*





24. Skip the *Custom Report View* page by clicking *Next*.
25. The report should be successfully validated. Click *Save*. You have now created your first report!



26. Now, you should see your new JasperServer Report within the Repository Browser screen. Click on the *new\_simple\_report* to view the report.

The next section shows how to create a more complex report that has a full set of input controls as well as scriptlet class JAR file dependencies.

### 4.3 Creating a complex report

You will now create a more complex report that contains several input parameters. To create the report, you must have access to all resources, including the *SalesByMonth.jrxml* file, the *SalesByMonthDetail.jrxml* file (the subreport), a data type, and an image for the logo.

**Note:** It is assumed you are very familiar with the *SalesByMonth.jrxml* file and the resources it requires.

*SalesByMonth.jrxml* and the resources it needs can be found in the *samples* directory included in the JasperServer Installers. After installation, the *samples* directory is located in the root of the installation directory. Additionally, a *samples.zip* is available for download from the *jasperintel.sourceforge.net* site.

Now, you'll use the available resources to create a report that behaves just like *SalesByMonth*.

#### 4.3.1 Naming the new report

1. In the repository browser, select the *MyReports* folder that you have created.
2. Click the Add Resource button in the toolbar and select JasperServer Report.
3. The Create Report Wizard appears.
4. For Name, enter *new\_report*.
5. For Label, enter *New Report*.
6. For Description, enter *This is a test report*.
7. Click Next.

### 4.3.2 Add the JRXML file, logo, and data source

8. Now that you've named the report, you'll add the resources you need to run it. Under *Locate JRXML File:*, click *From file System*.
9. Search on your file system for the *SalesByMonth.jrxml* file. (You should find it in the <install-dir>/samples/reports directory if you used the JasperServer installer. Otherwise, you can go to the [jasperintel.sourceforge.net](http://jasperintel.sourceforge.net) site and download the samples.zip file.)
10. Click *Next*.
11. The next screen shows you that you need the following resources: a subreport (the *SalesByMonthDetail.jrxml* file) and an image (a logo).
12. Click *Add Now* to the right of the *SalesByMonthDetail* resource name.
13. Click *Upload From File System* and browse for the *SalesByMonthDetail.jrxml* file.
14. Click *Next*.
15. Leave the name as *SalesByMonthDetail*. (Note that spaces are not allowed in the name field.)
16. For the Label, enter *Sales By Month Detail*.
17. Click *Next*. You'll see that you still have to locate the image for the logo.
18. Click *Add Now* to the right of the *Logo* resource name.
19. Click the *From Content Repository* radio button.
20. Use repository file URI */images/JRLogo*.
21. Click *Next*.
22. Leave the name *Logo* for name, label, and description.
23. Click *Next*. You'll see that you've successfully located all necessary resources.



### 4.3.3 Adding input controls

Now, continuing with the same report, you'll add the input controls.

The *SalesByMonth.jrxml* file has four input controls: *TextInput* (an integer), *CheckboxInput* (a Boolean), *ListInput*

(a single-select list), and DateInput (a date). You must use these exact names when creating the input controls.

25. Click *Add Control*.
26. Click the *Locally Defined*.
27. Click *Next*.
28. For Name, enter *TextInput*.
29. For Label, enter *Text Input Control*.
30. For Type, select *Single Value*.
31. Click *Next*.
32. Now you must define the data type (validation criteria) for the input control. Click *Locally Defined*.
33. Click *Next*.
34. For Name, enter *integer\_type*.
35. For Label, enter *Integer Control*.
36. For Type, select *Number*.

**Note:** In this example, you will not fill in most of the values on this screen. See *More on Datatypes* below for more information on this screen.

37. Click *Save*. You are sent back to the Resources List Screen in the report wizard. You should see *TextInput* in the list of resources.



38. Click *Add Control*.
39. Click the *Locally Defined*.

40. Click Next.
41. For Name, enter *CheckboxInput*.
42. For Label, enter *Checkbox Input Control*.
43. For Type, select Boolean.
44. Click Next. You should see the *CheckboxInput* in the list of resources.



45. Click Add Control.
46. Click the *Locally Defined*.
47. Click Next.
48. For Name, enter *ListInput*.
49. For Label, enter *List Input Control*.
50. For Type, select *Single Select List of Values*.
51. Click Next.
52. Click *Locally Defined*.
53. Click Next.
54. For Name, enter *list\_type*.
55. For Value, enter *List Type*.
56. Click Next.
57. Under Label, enter *First Item*, under Value, enter *1*. Click the plus button (+).

58. You are presented with a new row for Label and Values. Enter *Second Item* and 2 and click the plus button (+).
59. Lastly, add *Third Item* and 3 and click the plus button (+).
60. Click Save. ListInput appears in your list of resources.



61. Click Add Control.
62. Click the *Locally Defined*.
63. Click Next.
64. For Name enter *DateInput*.
65. For Label, enter *Date Input Control*.
66. For Type, select *Single Value*.
67. Click Next.
68. Click the *From the Repository* radio button.
69. From the drop down list, select */datatypes/date*.
70. Click Next.



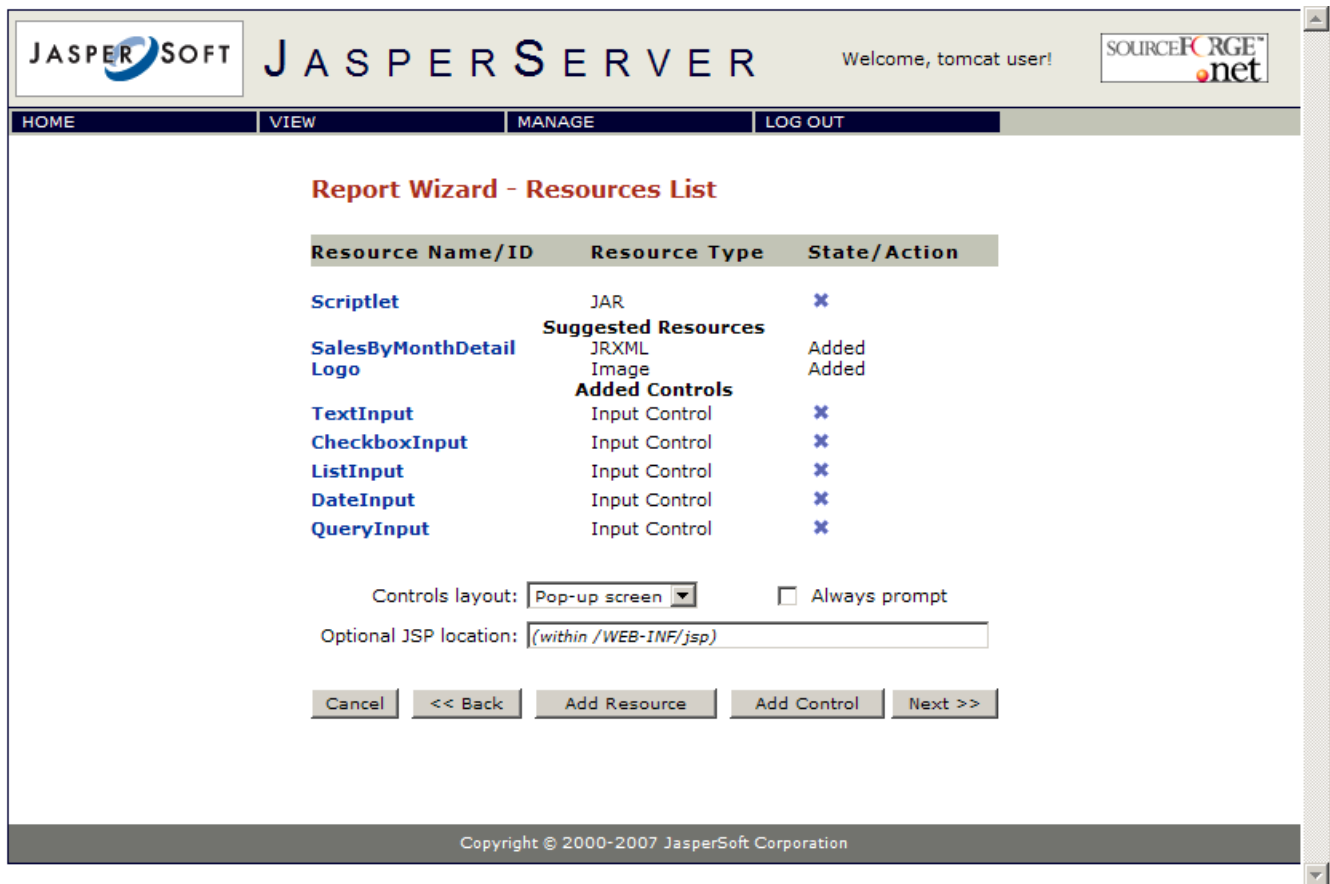
71. Click Add Control.
72. Click the *Locally Defined*.
73. Click Next.
74. For Name enter *QueryInput*.
75. For Label, enter *Query Input Control*.
76. For Type, select *Single Select Query*.
77. Click Next.
78. Click *Locally Defined* for the location of the query.
79. For query name and for label enter *testQuery*
80. Click Next.
81. Click *None* for the query data source and then Next.
82. For Query language leave the default SQL and for the query text enter the following: SELECT user\_name, first\_name, last\_name FROM users
83. Click Next.
84. For Value column enter *user\_name*
85. Add *first\_name* and *last\_name* for Visible columns using the + button.
86. Click Save.
87. You are now done with input controls! Begin adding external resources below.

In the same screen, one can choose whether the report input controls are to be displayed in a pop-up screen on the "View Report" page or in a separate page. A custom JSP page to be used for displaying the input controls can also be specified here; the JSP file needs to already be present on the server.

#### 4.3.4 Adding external resources

The JRXML file contains two external resources: a JAR file scriptlet and a resource bundle for localization. This section shows how to include them in the report.

- Click Add Resource.
- Choose *From File System* and browse for the scriptlet.jar file (samples/jars). Click Next. A screen appears saying the scriptlet was successfully loaded.
- For Name, enter *Scriptlet*. (This name is specified in the .jrxml file, so you must use this name.)
- For Label, enter *Scriptlet JAR*.
- Click Next.



- Now you'll upload the resource bundle. Click Add Resource.
- From the file system, browse for *sales.properties* (samples/resource\_bundles) and click Next. A screen appears saying *sales.properties* was successfully loaded.
- For Name, enter *sales.properties*. (You must use the same name as the file specified in the *SalesByMonth.jrxml* file.)
- For Label, enter *Sales Properties*.

- Click *Next*.
- You have returned to the Resource List screen. Continue to the next section to complete your report.

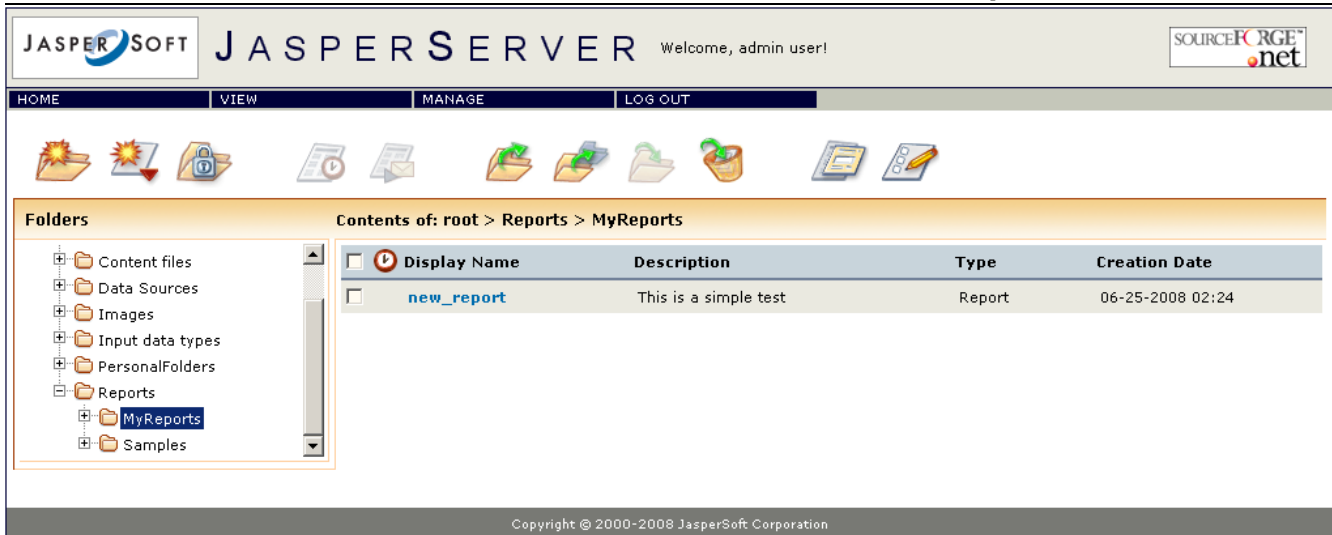


### 4.3.5 Adding a datasource

You have now finished adding all the report's resources. Now you'll add a datasource and validate the report.

75. On the Resources List screen, click *Next*.
76. Click the *From the Repository* radio button.
77. From the drop-down list, choose */datasources/JServerJdbcDS* (default selection).
78. Click *Next*. A screen appears to say the report was successfully validated.
79. Click *Save*. The report appears as published in the repository.





You can now click on your new report to view it. When you enter values in the the input control fields, these values appear toward the top of the report. Input Controls are typically used to modify runtime selections such as WHERE clauses in SQL queries.

### 4.3.6 More on data types

A data type in JasperServer is more than just a classification of the data. It also includes validation criteria. The data type description page contains the following fields:

<i>Field name</i>	<i>Description</i>
<b>Name</b>	A name of your choosing for the data type.
<b>Label</b>	A label of your choosing for the data type.
<b>Description</b>	Any additional information you want to provide about the data type.
<b>Type</b>	The data type, in a stricter sense. Possible values are <i>Text</i> , <i>Number</i> , <i>Date</i> , or <i>Date-Time</i> .
<b>Max length</b>	The maximum number of characters in the field.
<b>Decimals</b>	The number of decimal places in the field.
<b>Pattern</b>	A regular expression that restricts the possible values of the field.
<b>Minimum value</b>	The lowest permitted value of the field.
<b>Is strict minimum</b>	If checked, the minimum value itself is not permitted; only values greater than the minimum value are permitted.
<b>Maximum value</b>	The highest permitted value of the field.
<b>Is strict maximum</b>	If checked, the maximum value itself is not permitted; only values less than the maximum value are permitted.

## 4.4 Creating reports using the iReport plugin

The JasperServer plugin for iReport provides an easy and quick way to manage JasperServer Reports inside the JasperServer repository. The plugin uses a set of web services to interact with the server, and it is able to handle more than one server at the same time.

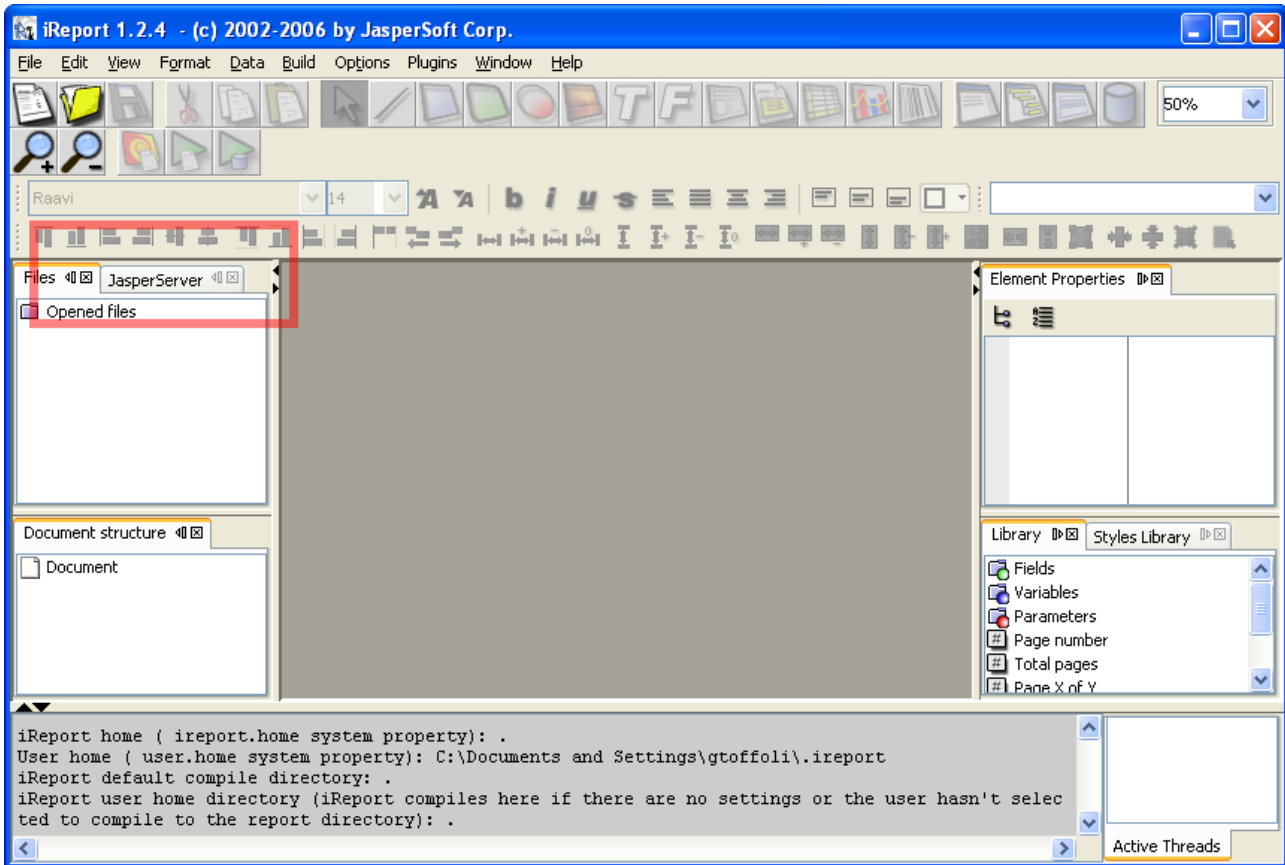
### 4.4.1 Plugin features

- Repository browsing

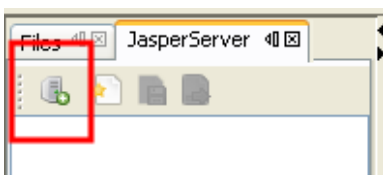
- Create / modify / delete folders
- Create / modify / delete generic resources (images, fonts, jrxml files, jars, property files, ...)
- Create/Modify/import datasources (JDBC, JNDI, JavaBean)
- Create / modify / delete JasperServer Reports
- Create/modify/delete datatypes
- Create/modify/delete List of values
- Create/modify/delete controls
- Add/remove/link controls to/from JasperServer Reports
- Drag and drop of images and subreports into the design window
- Repository resource chooser dialog

### 4.4.2 Making connection to JasperServer

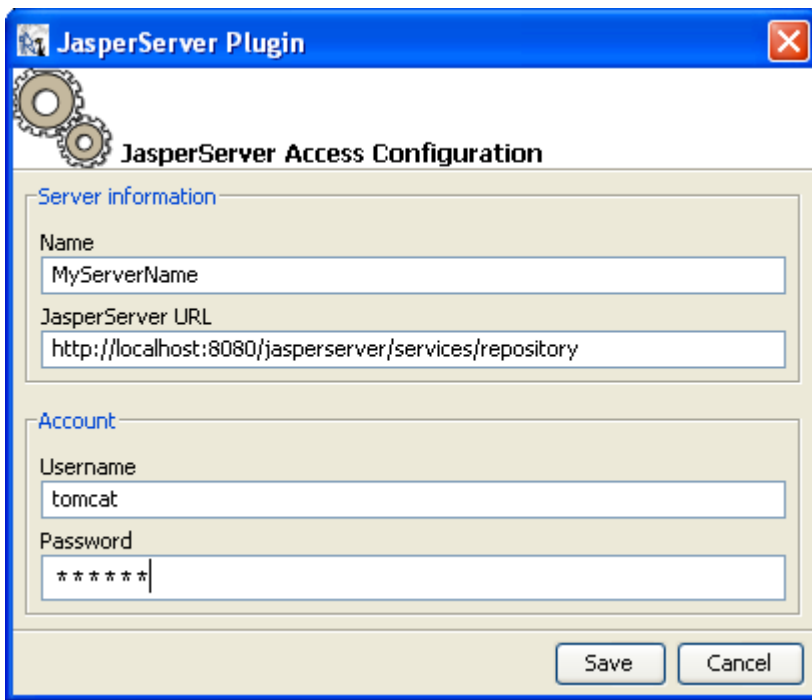
For Windows platform, select the Start/All Programs/JasperSoft/iReport-1.3.x/iReport-1.3.x menu item to start iReport. (See 5.3 for a detailed description of installing and configuring the iReport plugin.)



When the plugin is started for the first time, no servers are yet configured. To add a server, press the first button in the plugin toolbar.



You get a dialog box titled "JasperServer Plugin" that allows for JasperServer access configuration. In this dialog box, you will enter the information that will enable you to connect to your JasperServer Web Services application.



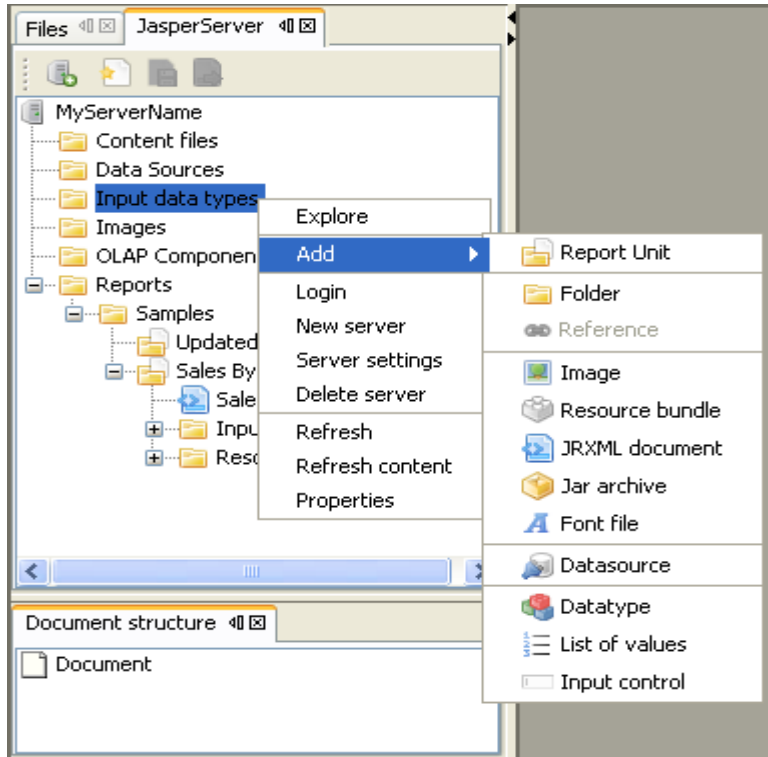
For instance:

Name:	MyServerName
JasperServer URL:	<you should only need to change the hostname>
Username:	<jasperadmin>
Password:	<password>

A server entry should now appear in this pane.

### 4.4.3 Using the plugin

Double-click on the server entry to see a list of Resources at the root of the JasperServer repository. You may view the property information on the Resources in the repository. You may download, edit and upload jrxml files. You may download images and other files (using right-click "export file").



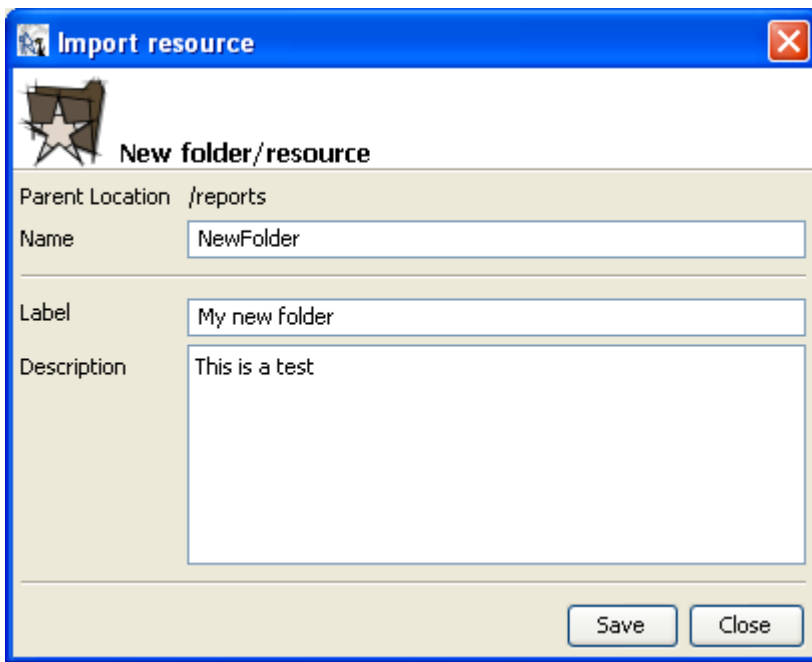
You can browse the repository double-clicking the server and folder icons.

### 4.4.4 Folders

When a folder is selected, you can add one of the following objects into a folder:

- another folder
- a brand new JasperServer Report
- a generic resource (like an image or a JRXML file)
- a datasource definition
- a datatype
- a list of values (used inside a control)
- an input control

To create a folder, select the parent folder (or the server entry to create a folder in the root), and click Add / Folder.

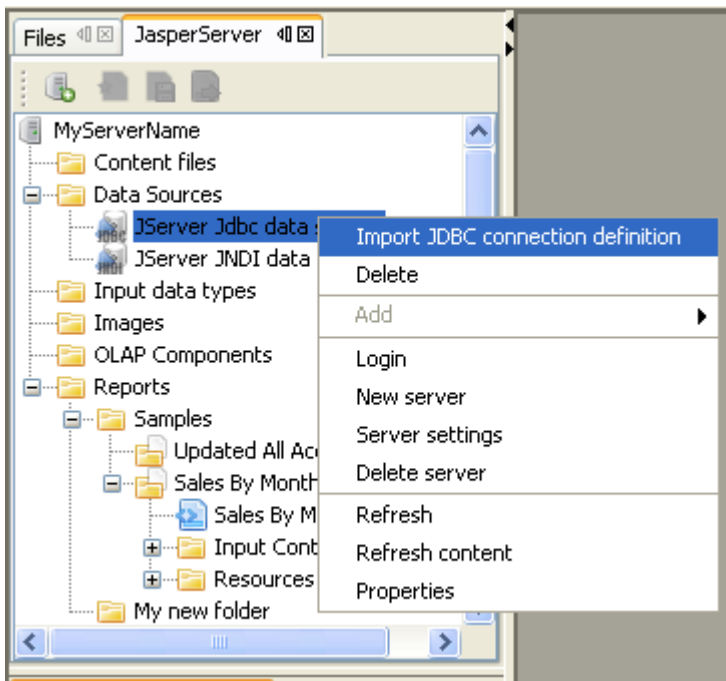


Only the name and the label are mandatory fields. To delete a folder select the “Delete” from the popup menu.


#### 4.4.5 Importing JDBC connections into iReport

You can import a JDBC datasource from the server into iReport selecting the JDBC datasource from the repository tree. Right-click on the datasource, and select the item Import JDBC connection definition.

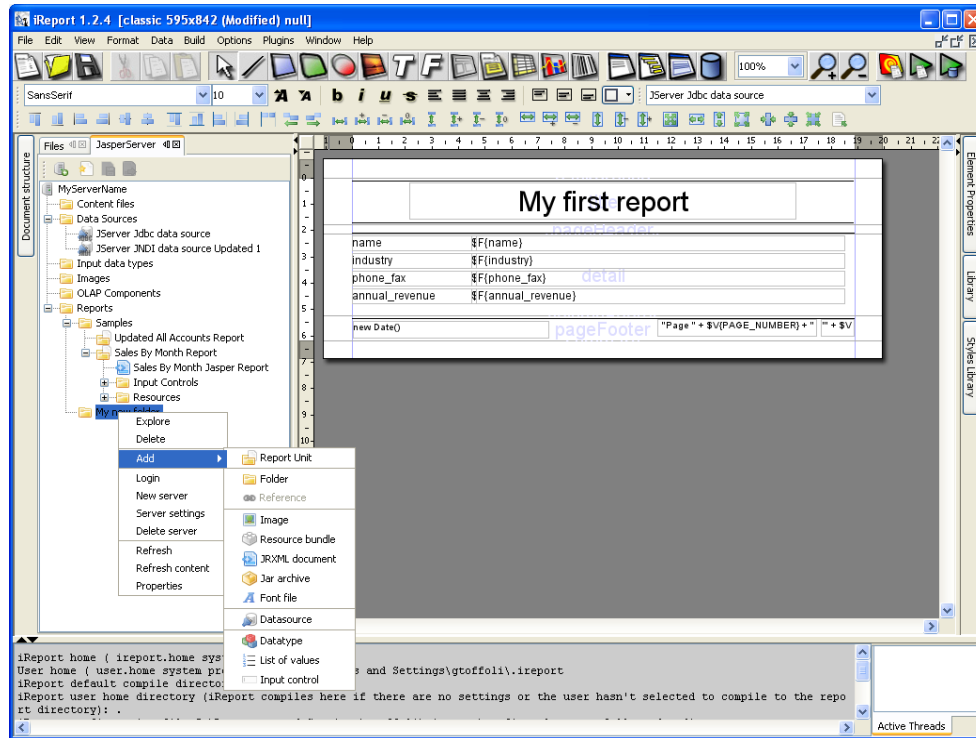
The new connection will stored in the iReport datasources/connections list. Now you can use this new connection with the iReport wizard to create your reports.



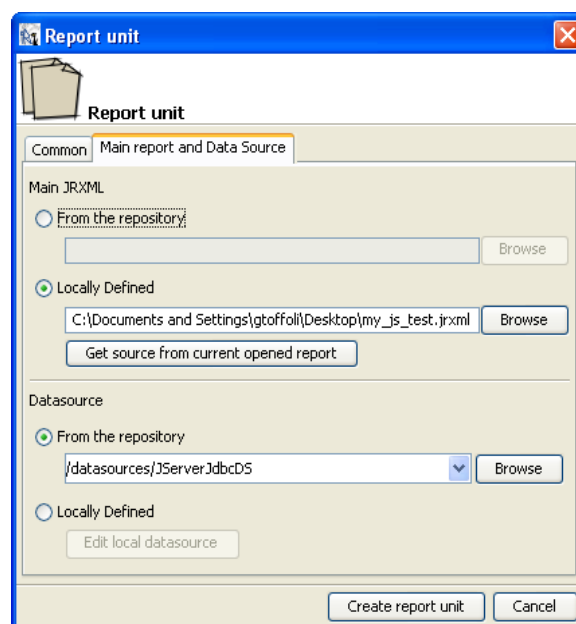
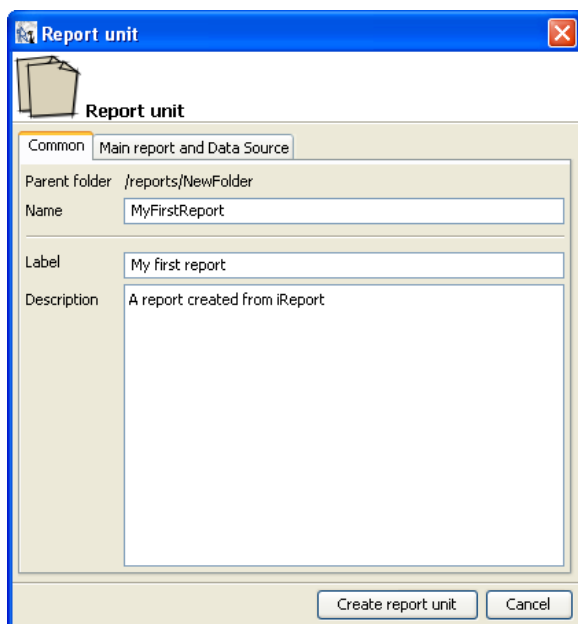
### 4.4.6 Creating a JasperServer Report from iReport

JasperServer Reports are displayed as special folders with the following icon: 

To create a brand new JasperServer Report, you need at least a JRXML file (it will become the main JRXML for the JasperServer Report).



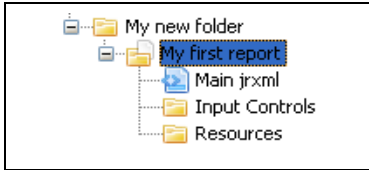
Choose the parent folder for the new JasperServer Report, select Add / JasperServer Report. You'll get the JasperServer Report dialog.



The dialog is composed by two tabs: “Common” and “Main Report and Datasource”. All fields, except Description, are mandatory.

If you want import the current opened file, save it before creating the report, and press the button “Get source from current opened report”.

If you want, you can pick an existing JRXML file from the repository, and/or define a local datasource (that is belonging to the JasperServer Report).




In any given JasperServer Report, there is always a main jrxml file, a set of input controls (zero or more) and a set of resources (zero or more).

### 4.4.7 Edit a JRXML

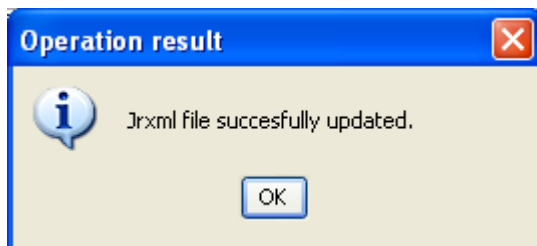
To edit a JRXML, double click on the JRXML file entry in the repository tree. If you have created a JasperServer Report from the current opened report, close this report, because this IS NOT the report present now on the server.

When you double click on the JRXML file, the JRXML is extracted from the repository and stored in a local copy into the directory <USER\_HOME>/.ireport/jstmp. This directory is never clean.


The JRXML source is automatically opened in iReport.

To restore the modified file on the server, save it, then select the file entry in the repository tree, and click the button  on the plugin toolbar.

A message will confirm that the operation was completed successful.

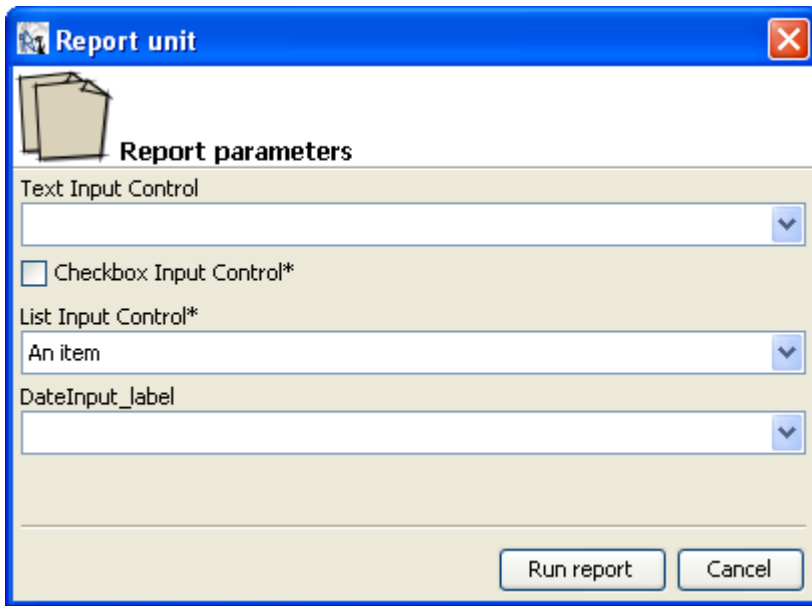


### 4.4.8 Running a JasperServer Report from iReport

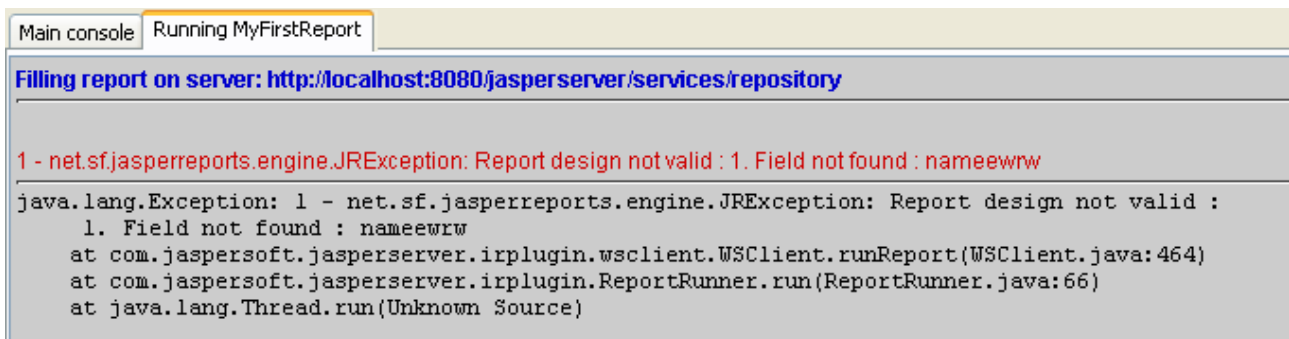
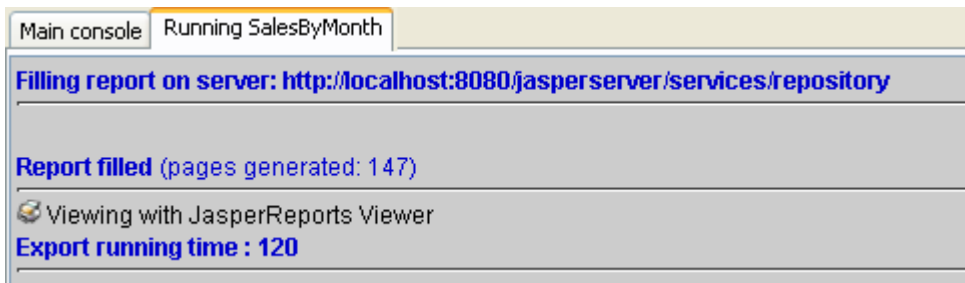
To run a JasperServer Report on the server from iReport using all export options set in iReport, select a JasperServer Report from the repository tree, and click the button  on the plugin toolbar.

If the JasperServer Report has input controls, the report parameters window will ask for values.



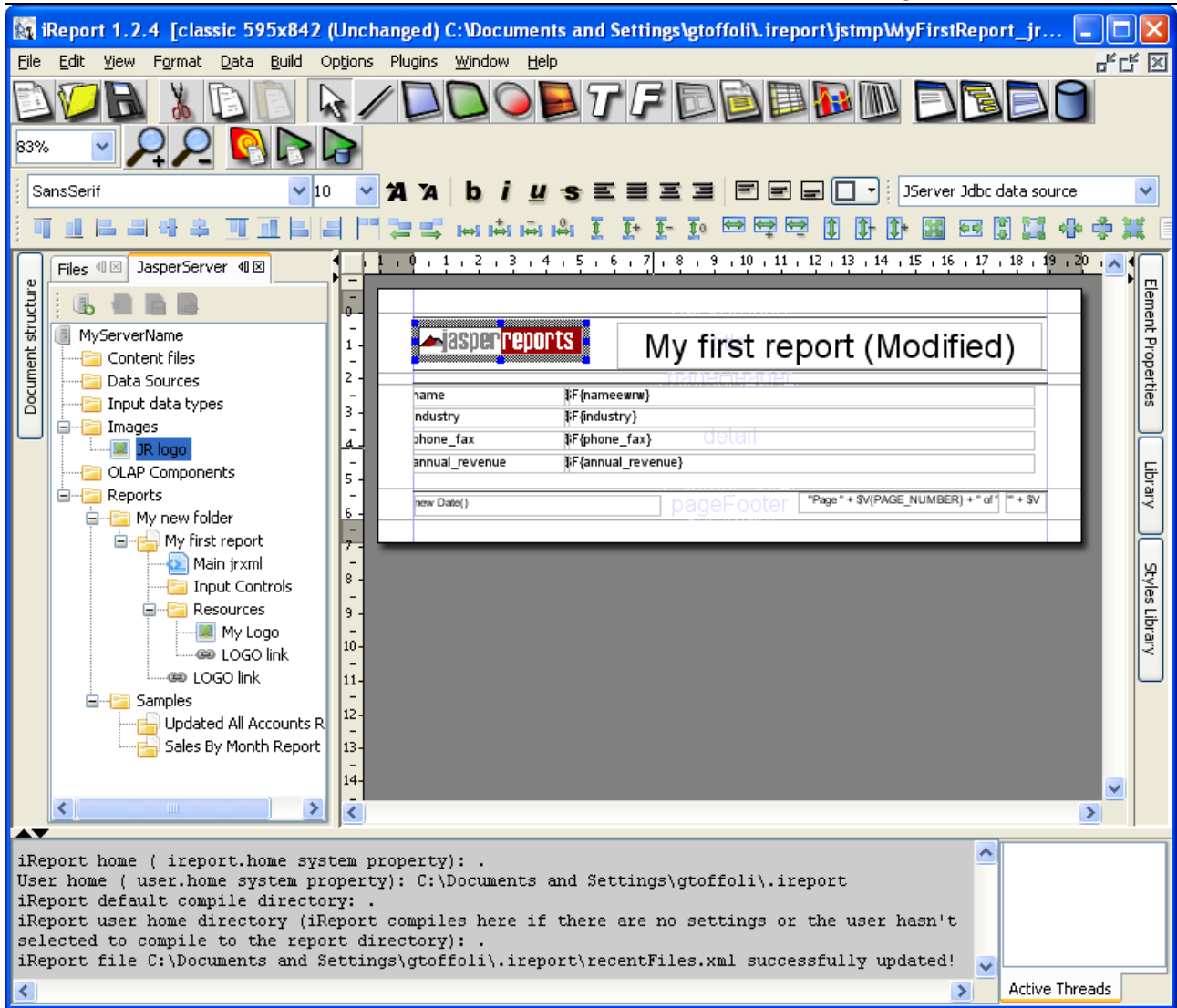


All messages coming from the server will be displayed on the iReport log pane.



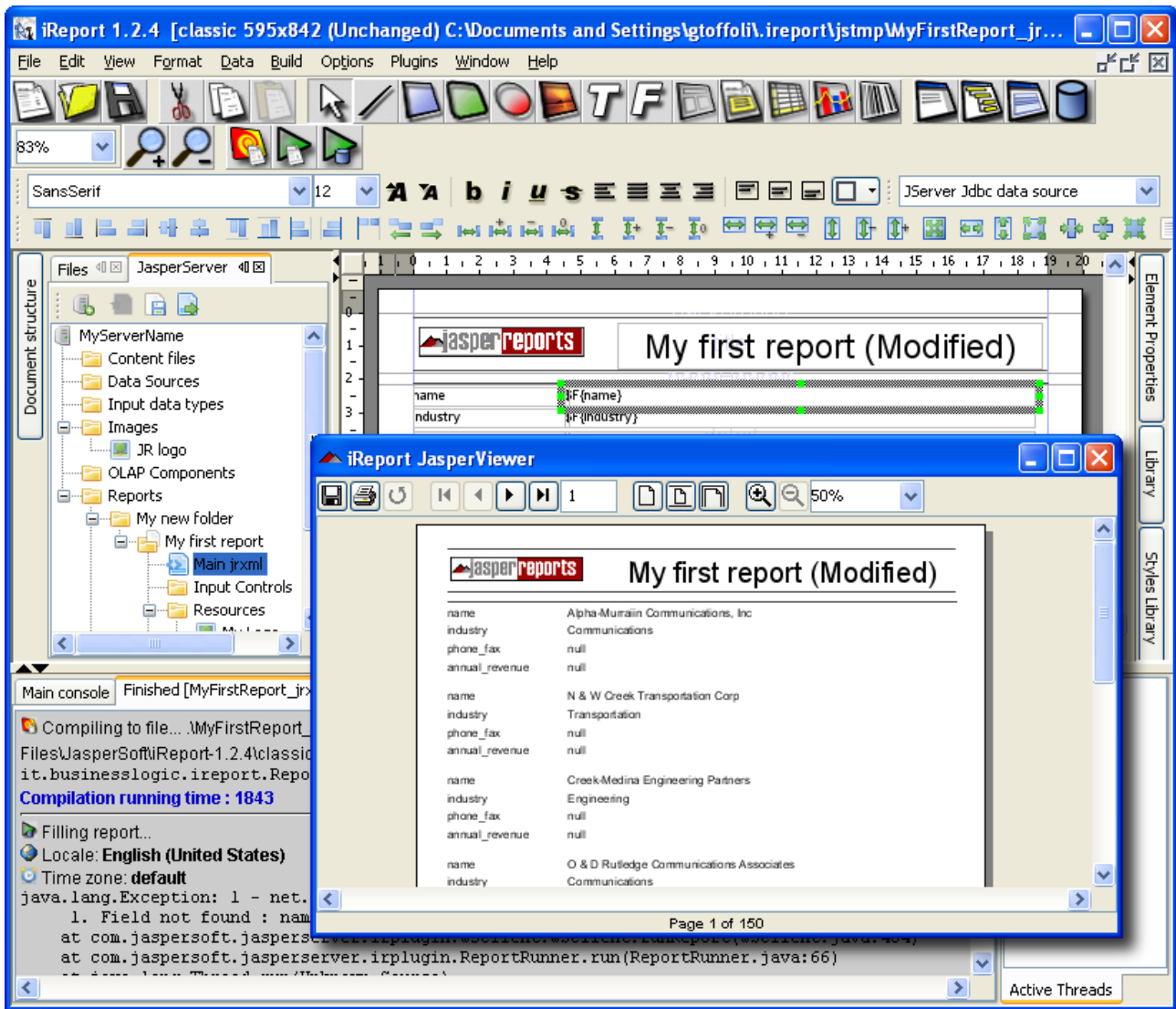
#### 4.4.9 Adding an image to a report

If you want add an image stored in your repository into a report, you can drag the image into the design window. The plugin will add the image element for you, setting the right expression to refer the image into the server repository.



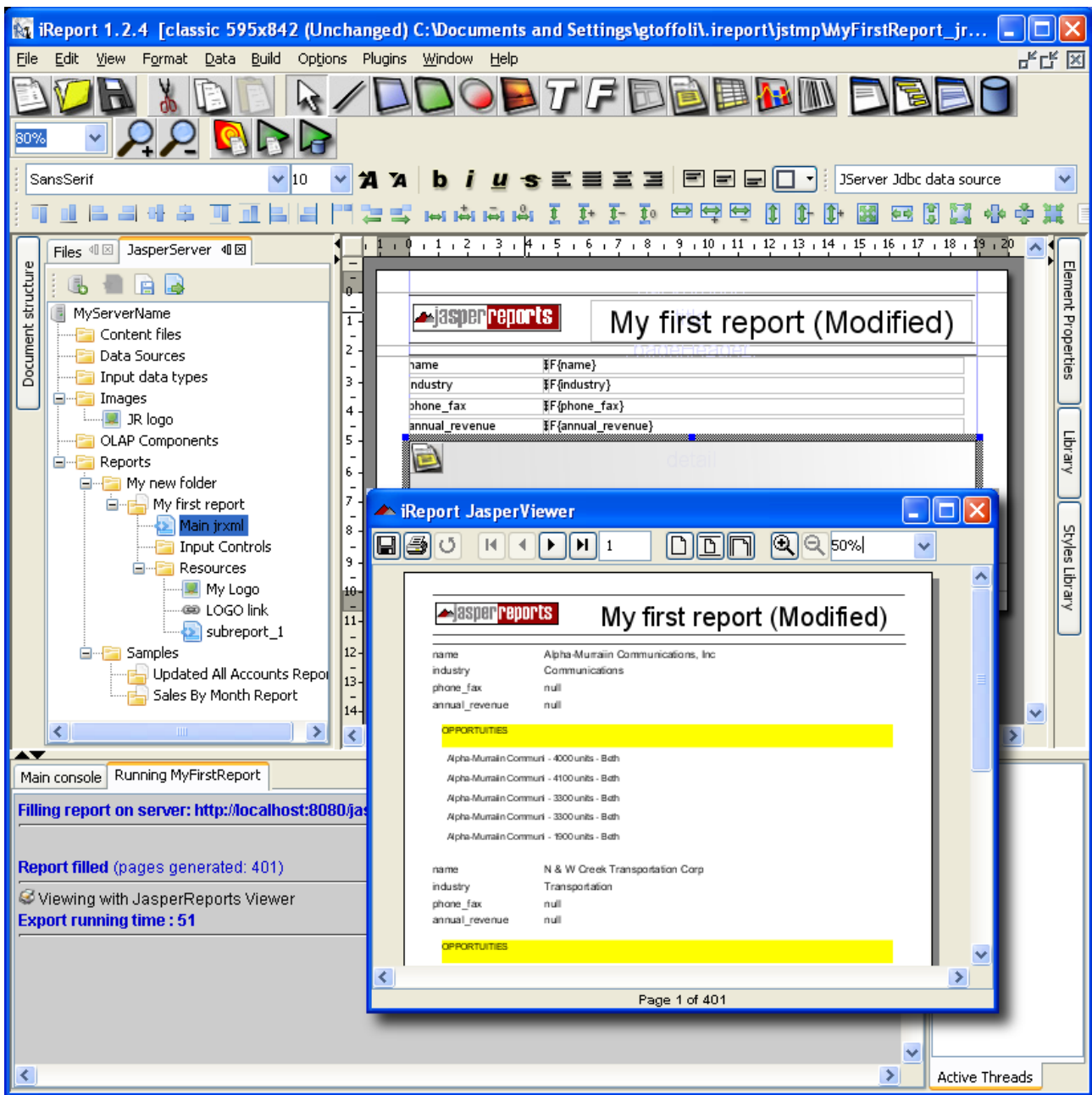
74

Save and run the report as described in the previous paragraph to see the result.



#### 4.4.10 Adding an subreport to a report

Similarly to what we have done to add an image, we can add a subreport to a report. First of all we have to create the JRXML that will be used as subreport, then we will add the new JRXML as resource to the repository, or to the JasperServer Report.



There is no way to try a subreport on the server, you can only try the whole report.

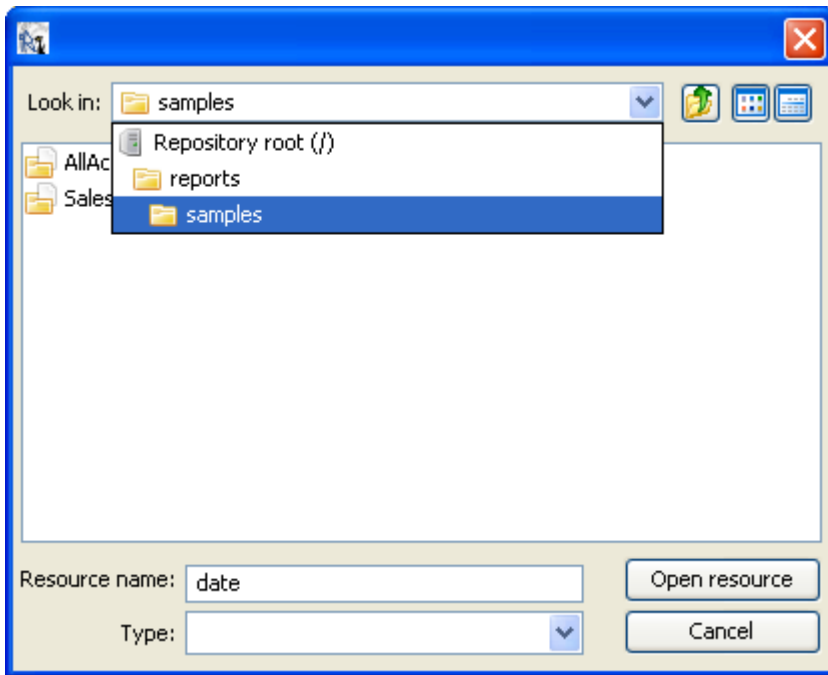
Edit the main jrxml and drag into the detail band the subreport jrxml entry. A subreport element will be created. The main report uses a JDBC connection so you have to set the connection expression of the subreport to `#{REPORT_CONNECTION}` in the element properties dialog, subreport tab.

#### 4.4.11 Resources, input controls, data types, lists of values and data sources

The plugin provides a way to manage resources like images, jars, jrxml files, properties file for localizable reports, etc..., input controls, data types, lists of values and data sources. The user interface provided by iReport to deal with these objects is similar to the one provided by the web UI. Refer to the sections below for the meaning of each field requested creating or managing a specific resource.

#### 4.4.12 Repository resource chooser

To locate a resource in the repository (i.e. defining a JasperServer Report that uses a JRXML already stored on the repository), the plugin provides a special component similar to a file chooser dialog.



Use it to easily navigate the repository and pick the resource your want.

#### 4.5 *Adding resources directly to the repository*

The example in this document shows how to create a complete report and place it into the repository. You can also use the Add Resource button in the repository browser to configure the following individual resources and add them to the repository for future use:

- Data Source
- Query
- Input Control
- Data Type
- JRXML
- Image
- Font
- JAR
- Resource Bundle

To add any of these resources to the repository:

1. Navigate to the folder where you want to put the resource.
2. Choose the resource type from the drop down list next to the Add button.
3. Click Add
4. Follow the steps in the resulting screen(s).
5. When you complete the screen(s), click Save. The resource is added to the repository.

## 4.6 Move/Copy resources in the repository

The repository browser UI (and the JasperServer API in general) allows moving and copying folders and resources within the repository.

### 4.6.1 Copying resources

In the following example we are going to see how to copy an existing report into a destination folder:

1. Navigate to the folder where the source report definition is found.

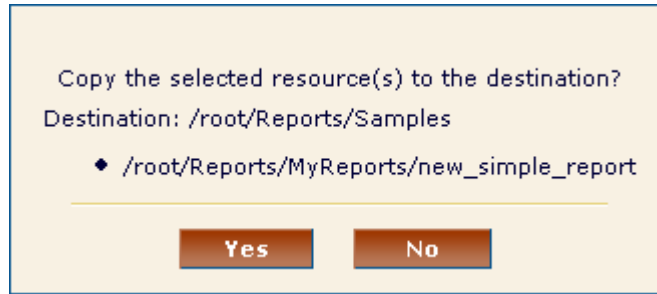


2. Select the report definition (and the other resources in that folder that you want to copy at the same time). In our case, the resource to copy is the *new\_simple\_report* created earlier in this chapter.
3. Click the *Copy* button in the toolbar or select the *Copy* option on the right click of the mouse on the chosen resource.
4. The mouse pointer changes to reflect we are during a copy operation.
5. Navigate to the destination folder (the *Samples* folder in our example).



6. Click the *Copy Here* button in the toolbar or select the *Copy Here* option on the right click of the mouse

on the destination folder in the report browser tree view.



7. Confirm the copy operation by clicking OK in the pop-up dialog that appears.
8. The report copy should appear in the destination folder.

### 4.6.2 Moving a folder

Moving content is very similar to copy, and we are going to see now how we can move a folder and its content to another location within the repository:

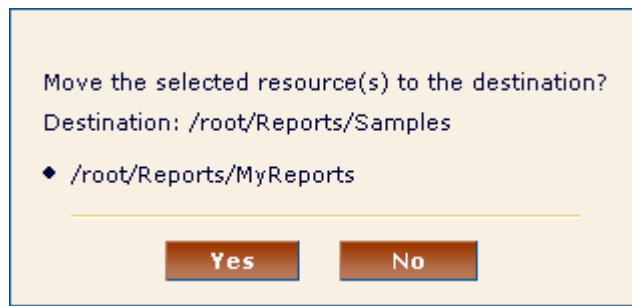
1. Navigate to the folder that you want to move, in our case the *MyReports* folder created earlier in this chapter.



2. Click the *Move* button in the toolbar or right click on the target folder and select the *Move* option in the pop-up menu.
3. Navigate to the destination folder.



- Click the *Move Here* button in the toolbar or select the option from the right mouse click on the destination folder.



- Confirm the move operation by clicking OK in the pop-up dialog.
- The folder should appear in the destination folder and should have disappeared from its former parent folder.

## 4.7 Security Maintenance

### 4.7.1 Overview

JasperServer has an integrated security framework that operates on several levels, namely:

- Authentication: identifying users
- Authorization
  - Access to screens
  - Menu options
  - Objects in the metadata repository

#### 4.7.1.1 Authentication

In order to customize the user experience and restrict access to resources in the system, users need to be authenticated to identify them and their user profiles, including their roles within the system. JasperServer tracks users with its metadata repository. The metadata repository can be the authentication source, i.e., Definition and management of user passwords, or we can also authenticate against a variety of authentication services, such



as LDAP (i.e., Microsoft Active Directory, Novell eDirectory). The open source Acegi security framework that is used has many configurable options for:

- Single sign-on: JA-SIG's Central Authentication Service (CAS)
- Java Authentication and Authorization Service (JAAS)
- Container security – Tomcat, Jetty
- SiteMinder

Encryption and authentication standards are also supported.

- HTTPS: including requiring HTTPS
- HTTP Basic
- HTTP Digest
- X509

The Acegi framework is readily extensible to integrate with custom and commercial authentication services and transports.

Authentication occurs by default through the Web user interface, forcing login, and also through HTTP Basic authentication for Web Services, such as iReport and for XML/A traffic. JasperServer will synchronize automatically with an external authentication service without the external users needing to be created in JasperServer first. Roles passed in with the externally authenticated user are included in the JasperServer metadata.

#### **4.7.1.2 Authorization**

With the user's identity and roles established, JasperServer controls what the user can do and see in a variety of ways.

- Menu options

Menu options appear depending on the user's roles. Normal users do not have access to administrative functions. This can be modified through reconfiguration. See the Developer's Guide for details.

- Screen access

Individual screens can be blocked for access by people with given roles. For example, the default security framework only allows administrators access to repository maintenance screens. This configuration of what screens are available for what roles is done in configuration files. Assigning appropriate roles to individual users through the administrative user interface configures the precise screen access for users.

- Object level security

Objects in the repository and folders can have permissions set on them so that users without sufficient rights cannot see and access those resources. For example, you may have management reports that are not for all users. Permissions can be set to block access for only users with a management role. This filtering capability is used regardless of whether the user is going through the Web user interface or Web Services. See Setting Permissions below.

## **4.7.2 Creating a role**

Roles are sets of permissions that you can assign to users.

To create a role:

1. Click the Role menu.

2. Click Add Role.
3. Type the name of the role in the Role Name field.
4. Click Submit.

### 4.7.3 Creating a user

To create a user:

1. Click the Users menu.
2. Click Add User.
3. Fill in the User ID, Full Name, Email Address, Password, and Confirm Password fields.
4. Click Enable this user to allow this user to log in.
5. Click Submit.
6. Edit the user to select the roles that you want to assign to this user (ROLE\_USER will be assigned by default). If you want to assign other roles, click on the edit roles link. In the Edit Roles dialog select the desired roles from the *Available* list and click on the right arrow in order to assign them to the user.
7. If you want to remove some assigned roles, select them from the *Assigned* list and click on the left arrow to remove them
8. Click Done.

### Add User

**User ID (required)**

Must be unique within the organizations.  
The following characters are not supported: |, [ ], ` , " , ' , ~ , ! , # , \$ , % , ^ , & , [ , ] , \* , + , = , ; , : , ? , } , { , ( , ) , [ , / .

**Full Name**

**Email Address**

**Password (required)**

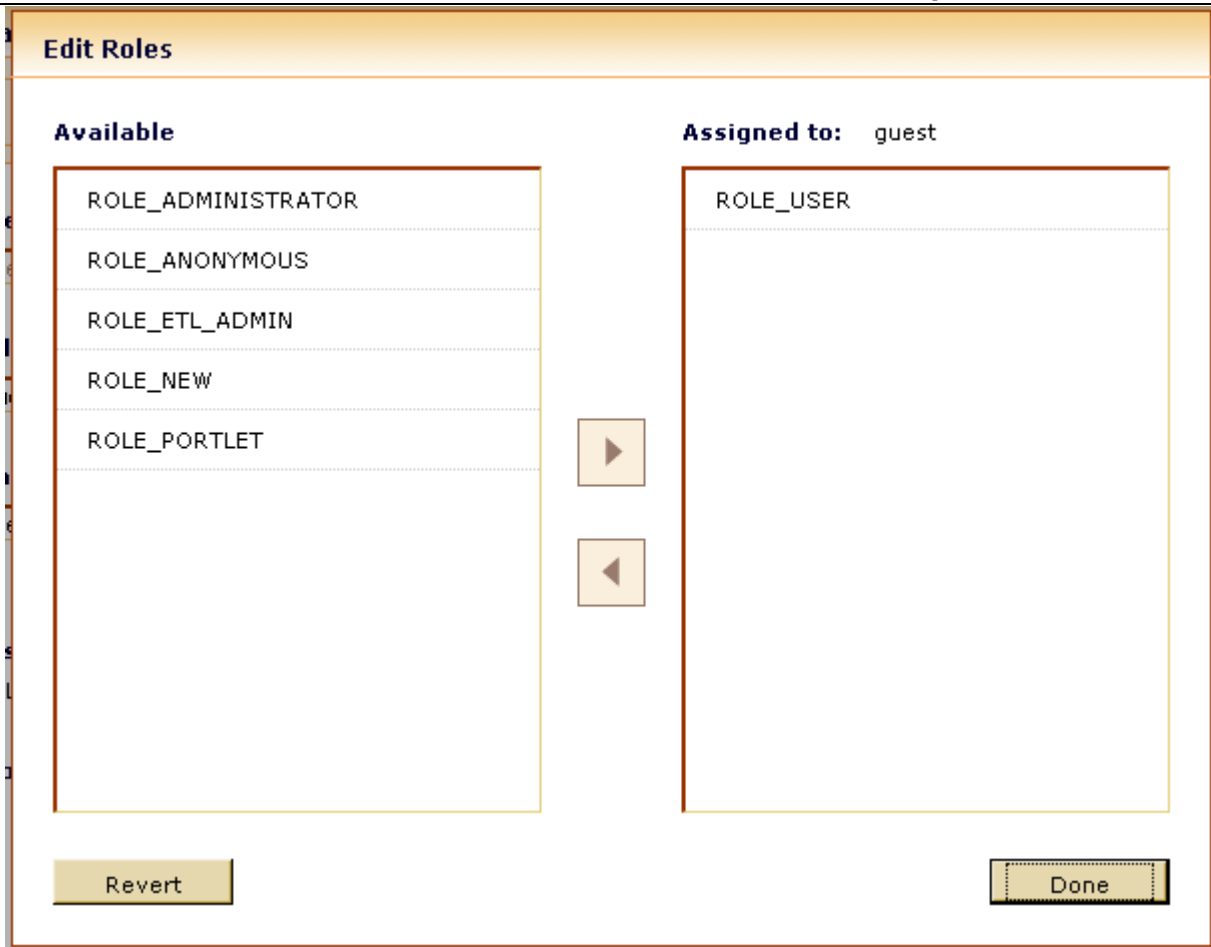
**Confirm Password (required)**

Enable this user

### Manage Users

Users	Details
admin	<input type="button" value="Save"/> <input type="button" value="Cancel"/> <p><b>User ID</b> <input type="text" value="guest"/></p> <p><b>Full Name</b> <input type="text" value="Guest User"/></p> <p><b>Email Address</b> <input type="text" value="guest@guest.com"/></p> <p><b>Assigned Roles (edit roles...)</b> ROLE_USER</p> <p><b>Profile Attributes</b></p> <p><b>Password</b> <input type="password" value="..."/></p> <p><b>Confirm Password</b> <input type="password" value="..."/></p> <p><input checked="" type="checkbox"/> Enable this user</p>
anonymousUser	
CaliforniaUser	
etladmin	
etluser	
<b>guest</b>	
jasperadmin	
joeuser	
tomcat	



#### 4.7.3.1 Changing role assignments for existing users

1. Click the Roles menu.
2. Select the role that you want to apply to an existing user and click Edit.
3. For users that you want to add to this role, click the arrow and move them from the *Available* list into the *Assigned* list.
4. For users that you want to delete from this role, click the arrow and move them from the *Assigned* list into the *Available* list.
5. Click Done.

**Assign Users**

**Available**

admin
anonymousUser
CaliforniaUser
etladmin
etluser
guest
jasperadmin
joeuser
tomcat

▶

◀

**Assigned to:** ROLE\_NEW

*No users found*

Revert

Done

#### 4.7.4 Configuring Password Options

JasperServer includes several password configuration options, described in the following sections.

##### 4.7.4.1 Changing Password Encryption Settings

By default, the passwords JasperServer stores in the database are not encrypted. If your security standards require you to encrypt passwords, you can configure JasperServer to handle encrypted passwords. These passwords include:

- JasperServer user passwords
- JDBC data source admin passwords
- XML/A connection admin passwords

Since JasperServer always includes at least one user (admin), you can't simply configure JasperServer to use encrypted passwords. To change the existing users, you must also use the import/export utility to extract them from the repository, then change the configuration, and finally re-import the users back into the repository. The import/export steps ensure that your users' passwords are properly encrypted.

This section describes the available password encryption configuration options:

Password Configuration Options		
Configuration File		
... \WEB-INF\applicationContext-security.xml		
Property	Bean	Description
passwordEncoder	daoAuthenticationProvider	Uncomment this property to enable the encryption of passwords.
allowEncoding	passwordEncoder	Determines whether JasperServer should encrypt the passwords it writes to the database. Set it to TRUE to use encrypted passwords
secretKey	passwordEncoder	The "salt" key to use as a pass phrase when encrypting passwords. This string is used to encrypt passwords. This value can be a simple string or a numeric representation that can be parsed by Integer.decode(). For example: String: This is my secret key Numeric representation: 0xC8 0x43 0x29 0x49 0xAE 0x25 0x2F 0xA1 0xC1
keyInPlainText	passwordEncoder	Determines whether the secret key is a simple string or a numeric representation. Set this parameter to TRUE if the secretKey is a string; set it to FALSE if the key is a numeric representation.
secretKeyAlgorithm	passwordEncoder	The name of the algorithm to use, such as DESede.
cipherTransformation	passwordEncoder	The name of the transformation, such as DES/CBC/PKCS5Padding.

JasperSoft recommends that you change the salt key in your production environment. This ensures that your encryption is unique to that environment.

The `secretKey`, `secretKeyAlgorithm`, and `cipherTransformation` must be consistent with each other. For example, if the `secretKeyAlgorithm` is `DESede`, the `secretKey` must be 24 bytes long. For more information about `secretKey`, `secretKeyAlgorithm`, and `cipherTransformation`, see Sun's `javax.crypto` documentation.

#### 4.7.4.2 Allowing Users to Change Their Passwords

In order to change their passwords, users click the *Change Password* link appears on the Login page. Enable this link only if JasperServer authenticates your users. If your users are externally-authenticated, do not enable the option. To allow users to change their passwords, you must edit a configuration file. Note that enabling the password expiration option (described in the following section) automatically enables the ability of end users to change their passwords.

Password Administration Option		
Configuration File		
... \WEB-INF\jasperserver-servlet.xml		
Property	Value	Description
allowUserPasswordChange	TRUE <any other value>	Set the value to TRUE to enable the <i>Change Password</i> link. Any other value disables it.

#### 4.7.4.3 Enabling Password Expiration

If your security policies require your users to change their passwords at regular intervals, you can enable password expiration. In this case, JasperServer prompts users to change their passwords at the interval you specify. For example, if you set the password expiry to 90 days, JasperServer prompts your users to change their passwords every three months. When a user's password expires, the user cannot log into JasperServer until she changes her password. The default value is 0; in this case, passwords don't expire.

When this option is enabled, JasperServer automatically enables the user password change options on the login page, even if allowUserPasswordChange is set to false.

Password Administration Option		
<b>Configuration File</b>		
...\\WEB-INF\\jasperserver-servlet.xml (controls the Login page) ...\\WEB-INF\\applicationContext-security.xml (controls web services)		
Property	Value	Description
passwordExpirationInDays	TRUE <any other value>	Set the value to any positive, non-zero value to specify the number of days after which a password will expire.

#### 4.7.5 Setting Permissions

In the repository browser for administrators, the "assign" permissions link is available for resources and users. Selecting this link takes you to:

##### Role Permissions Screen

Here you can set the level of permissions for the resource or folder for individual roles.

- None set: permissions have not been set for this resource or folder for the role. The permissions in effect for the role will be for the parent folder.
- No access: users with this role will not be able to access this resource or folder
- Read: Users with the role can read but not update the resource
- Administration: Users with the role can read, update and delete the resource

Select the level of permission and Update.

From the Role Permissions screen, the administrator can opt to set permissions at the user level for the resource.

## 5 Utilities

### 5.1 Import/Export

The import and export utilities let you extract resources from or add resources to a JasperServer repository. Import and export can be helpful when migrating between versions of JasperServer, or when moving between test and production environments.

#### 5.1.1 Import Resources

usage: ji-import [OPTIONS]

Reads a repository catalog from the disk (created using the ji-export command) and creates the named resources in the current JasperServer application repository.

**Options:**

<b>--prepend-path</b>	string to prepend to a URI path for all imported resources
<b>--update</b>	specifies that existing repository resources should be updated by catalog resources
<b>--input-dir</b>	path for import catalog directory
<b>--input-zip</b>	path for import catalog zip file
<b>--help</b>	print usage message

**Examples:**

- Import the `myDir` catalog folder, prepending `/importDir` to all repository URIs:  

```
ji-import --input-dir myDir --prepend-path /importDir
```
- Import the `myExport.zip` catalog archive file:  

```
ji-import --input-zip myExport.zip
```

**Notes:**

The `prepend-path` option is handy for avoiding URI path conflicts during import. For example, if the resource in the catalog file is `/images/JRLogo` and you set a `prepend-path` of `myNewDir`, then the resource is imported and created under the URI path `/myNewDir/images/JRLogo`. So, if you are importing a set of resources into a repository, and the URI naming conflicts with objects already in the repository, adding a `prepend-path` allows you to import the resources to a different location. However, this does not affect repository URIs in JRXML reports (for example, if a report image has `<imageExpression>repo:/images/JRLogo</imageExpression>`, that URI is not prepended during import).

During import, if a resource is found in the target repository that has the same URI as the resource that you are attempting to create and the “update” option was not used, the create operation is skipped, and the existing resource is left unchanged (no overwrite occurs); if the “update” option was specified and the type of the repository resource matches the type of the catalog resource, then the repository resource is overwritten by the catalog resource. Note that the “update” option results in all catalog resource indiscriminately overwriting repository resource, but it does not apply to other catalog objects such as users, permissions or report jobs.

**5.1.2 Export Resources**

usage: `ji-export [OPTIONS]`

Specifies repository resources such as reports, images, folders, users, roles, and scheduled jobs to export to a folder structure on disk.

The export output is known as a repository catalog.

**Options:**

<b>--uris</b>	Comma separated list of repository folder/resource URI paths
<b>--repository-permissions</b>	When this option is present, repository permissions are exported along with each exported folder and resource. This option should only be used in conjunction with <code>--uris</code>
<b>--report-jobs</b>	Comma separated list of repository report unit and folder URIs for which report unit jobs should be exported. If a folder URI is specified, the folder is recursively traversed and each report unit found has its jobs exported.
<b>--users</b>	Comma separated list of users to export; if no users are specified, all users are exported
<b>--roles</b>	Comma separated list of roles to export; if no roles are specified, all roles are exported
<b>--role-users</b>	When this option is present, each role export triggers the export of all users belonging to the role. This option should only be used in conjunction with



	--roles
--everything	Export everything: all repository resources, permissions, report jobs, users and roles. Equivalent to --uris / --repository-permissions --report-jobs / --users --roles
--output-dir	Output catalog folder
--output-zip	Output catalog zip file
--help	Print usage message

**Examples:**

- Export the /reports/samples/AllAccounts resource to a catalog folder:  
`ji-export --uris /reports/samples/AllAccounts --output-dir myExport`
- Export the /images and /fonts folders:  
`ji-export --uris /images,/fonts --output-dir myExport`
- Export all the resources with permissions to a zip catalog:  
`ji-export --uris / --repository-permissions --output-zip myExport.zip`
- Export all the resource and all the report jobs:  
`ji-export --uris / --report-jobs / --output-dir myExport`
- Export the report jobs of the /reports/samples/AllAccounts report unit  
`ji-export --report-jobs /reports/samples/AllAccounts --output-dir myExport`
- Export all the roles and users:  
`ji-export --roles --users --output-dir myExport`
- Export the ROLE\_USER and ROLE\_ADMINISTRATOR roles along with all users belonging to one of these roles:  
`ji-export --roles ROLE_USER, ROLE_ADMINISTRATOR --role-users --output-dir myExport`
- Export two users:  
`ji-export --users jasperadmin,joeuser--output-dir myExport`

**Notes:**

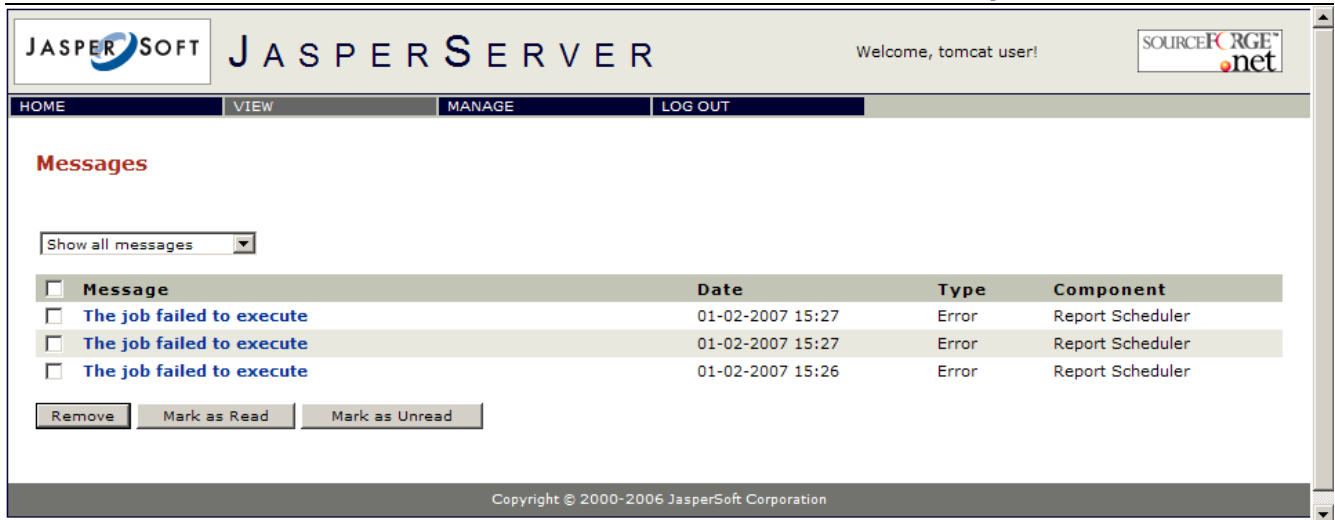
- The --uris option allows you to specify one or more resources or folder URIs. A URI can specify a resource such as a report. In this case, all associated resources such as images, subreports, data sources, resource bundles, and classfiles are exported. A URI can also specify a folder. If a folder is specified, the export operation exports all files and folders contained in the folder. In addition, it recurses through all its subfolders.
- When you export a user, in addition to the user information, the user's roles are also exported. When you import a user, if its roles exist in the repository, the user is given these roles.

**5.2 Message viewer**

The message viewer screens allows users to browse the server event log. This log is meant to be used by server component to log messages for events that require user notification. Currently the event log is exclusively used by the report scheduling module to log report execution errors.

**5.2.1 Messages list**

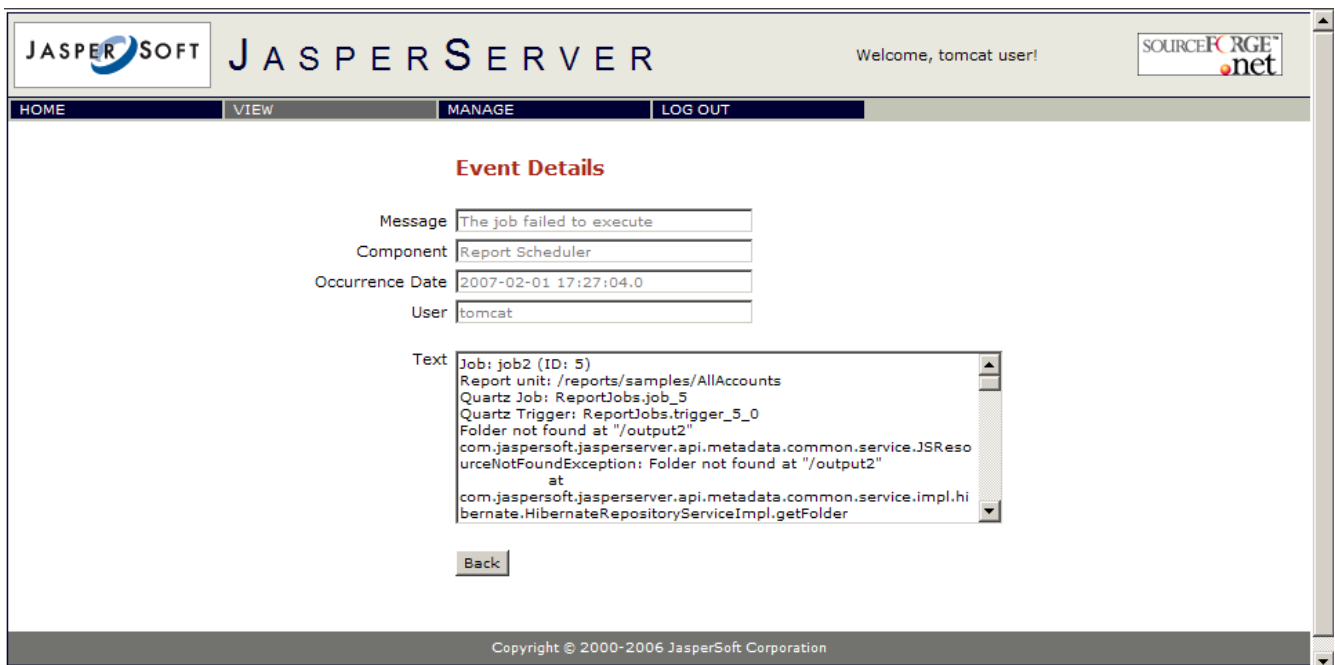
The *messages list* screen displays the list of events logged for the current user.



### 5.2.2 Message details

The full *message details* can be opened by clicking the event message.

The user can choose to browse all the messages or to see only the unread messages.



## 5.3 *iReport Plugin installation and configuration*

### 5.3.1 Requirements

- IReport 1.3.1 or greater
- Java Sun JDK 1.5 or greater (not mandatory but strongly suggested)
- JasperServer with enabled Web Services

### 5.3.2 Plugin installation

To install the plugin you have to drop into the *lib* directory of iReport the following jars:

jasperserver-ireport-plugin-1.1.0.jar

XmlSchema-1.0.2.jar

activation-1.1.jar

axiom-api-1.0.jar

axiom-impl-1.0.jar

axis2-1.0.jar

castor-1.0-xml.jar

commons-codec-1.3.jar

commons-httpclient-3.0.jar

mail-1.4.jar

neethi-1.0.1.jar

stax-api-1.0.jar

wSDL4J-1.5.2.jar

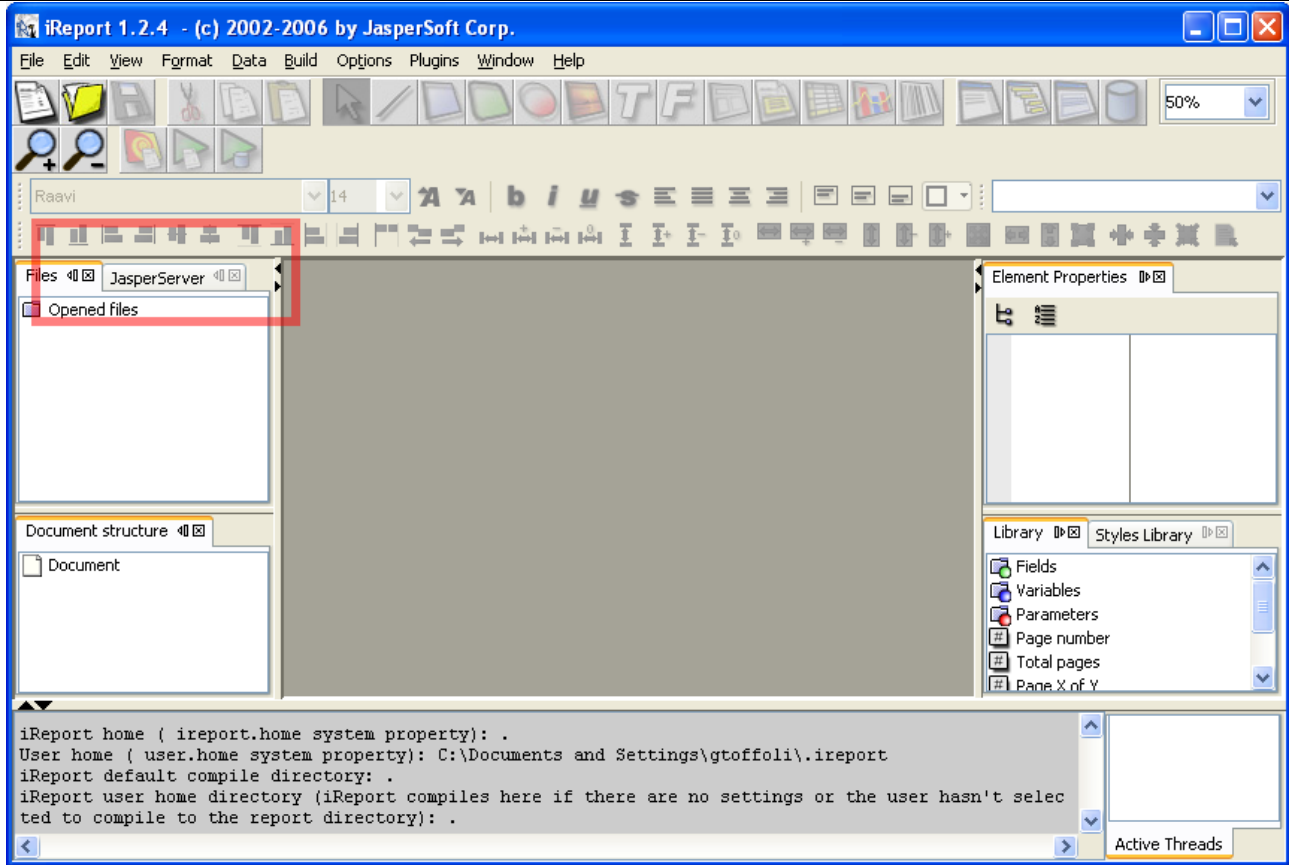
wstx-asl-2.8.2.jar

xbean-2.1.0.jar

All these jars are already in place if iReport is shipped in bundle with the JasperServer. iReport will load the plugin automatically when started. To launch iReport on the Windows platform, you may use the Start/All Programs/JasperServer/iReport menu item. On Linux simply type:

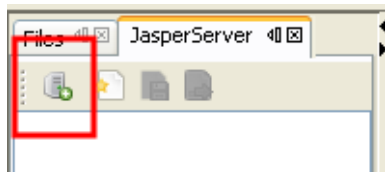
```
./iReport.sh
```

from the iReport home directory. You will see a JasperServer pane in the upper left hand side of your iReport workspace.

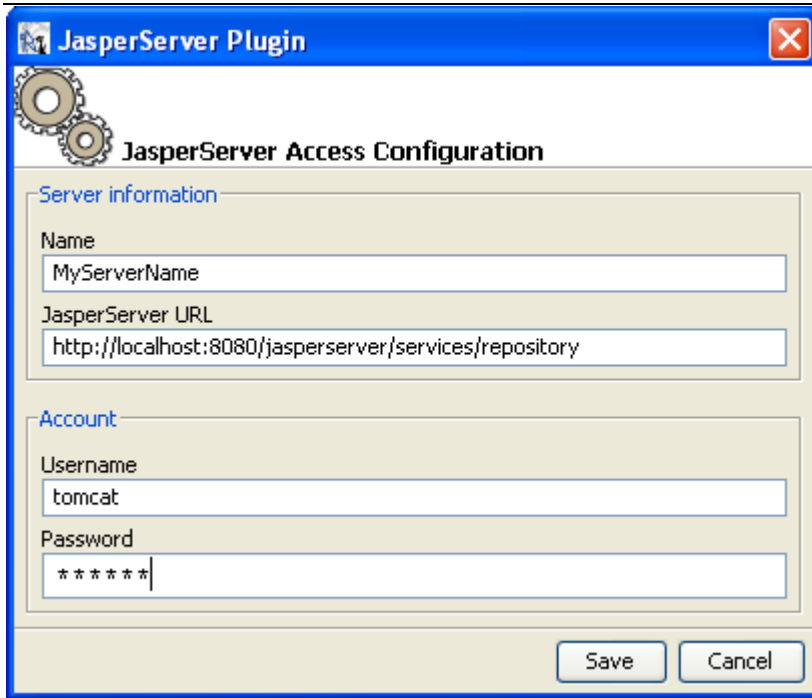


### 5.3.3 Configuring the plugin

When the plugin is started for the first time, no servers are yet configured. To add a server, press the first button in the plugin toolbar.



You get a dialog box titled "JasperServer Plugin" that allows for JasperServer access configuration. In this dialog box, you will enter the information that will enable you to connect to your JasperServer Web Services application.



The screenshot shows a dialog box titled "JasperServer Plugin" with a sub-title "JasperServer Access Configuration". It contains two sections: "Server information" and "Account".

**Server information**

- Name: MyServerName
- JasperServer URL: http://localhost:8080/jasperserver/services/repository

**Account**

- Username: tomcat
- Password: \*\*\*\*\*

Buttons: Save, Cancel

For instance:

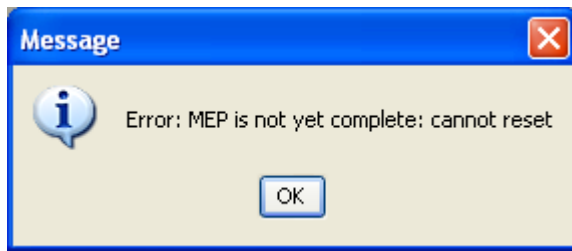
Name:	MyServerName
JasperServer URL:	<you should only need to change the hostname>
Username:	<jasperadmin>
Password:	<password>

A server entry should now appear in this pane.

Double-click on the server entry to see a list of Resources at the root of the JasperServer repository. You may view the property information on the Resources in the repository. You may download, edit and upload jrxml files. You may download images and other files (using right-click "export file").

### 5.3.4 Troubleshooting

If the connection fails, you can get a message similar to the following:



Check the login and password and be sure that the URL set for the server is exact. If needed, you can modify the server settings selecting it in the JasperServer pane and choosing the properties menu item from the popup menu.

## 6 Advanced configuration

This chapter presents some of the most common extension points that can be used to enrich or customize the JasperServer functionality and achieve the desired level of integration with the environment where it is deployed.

### 6.1 Custom Data Sources

JasperServer provides built-in support for many commonly used data sources, such as JDBC, Java beans, Mondrian, and XML/A. However, not all JasperReports data sources have corresponding JasperServer data sources, and you may have written your own custom JasperReports data source that you want to use within JasperServer. In either case, you will need to extend JasperServer to support additional data sources. Fortunately, the current version of JasperServer has a framework which makes it possible to add a new data source by adding just a few files to your JasperServer configuration.

This section will cover the following subjects:

- Instructions for installing examples of custom data sources in your JasperServer web application
- A description of JasperServer data sources and how they interact with JasperReports data sources
- Descriptions of all the components required to implement a new JasperServer data source
- Instructions on how to deploy your new JasperServer data source

#### 6.1.1 Background on data sources in JasperServer and JasperReports

A JasperServer data source is not the same thing as a JasperReports data source, but they work together closely. A JasperReports data source is an implementation of the `JRDataSource` interface which provides data organized in rows and columns to the JasperReports filler, which is responsible for producing a `JasperPrint` object. Each field declared in the JRXML corresponds to a column in the `JRDataSource` output.

A JasperServer data source is a persistent object in the JasperServer repository which stores properties which tell JasperServer how to create a `JRDataSource` (possibly in collaboration with a `JRQueryExecutor`, see below). These properties vary by the type of data source; for example, a JDBC data source will have a JDBC driver, URL, user, and password. A data source, like other repository objects such as data types and input controls, can be defined as a “public” repository object which can be used by any report unit (for example,

/datasources/JserverJdbcDS), or as a “local” object used by a specific report unit. Public and local data sources can be created and edited with the JasperServer web interface.

When JasperServer receives a request to run a report unit, it looks up the the report unit's data source, and maps that to an implementation of `ReportDataSourceService`, which is in turn responsible for handing back a `JRDataSource` based on the data source's persistent properties. The `JRDataSource` is used to fill the report and produce a `JasperPrint` object, which can be turned into HTML or any other supported output format.

Each JasperServer data source implementation has to do the following:

- Read and write persistent properties in the JasperServer repository
- Provide a user interface for creating and editing instances which is integrated in the JasperServer web interface
- Create a `JRDataSource` using the property values for a specific data source instance, or pass parameters to a `JRQueryExecutor` which produces the `JRDataSource`.

The existing data sources each require about a half dozen Java classes, along with many changes to Spring bean files, WebFlow configurations, message files, and JSP files. The custom data source framework described here provides the same functionality using a Spring bean file, a message catalog, and a minimum of one Java file (more for optional features).

### 6.1.1.1 Query executers

Query executers are implementations of the JasperReports interface called `JRQueryExecutor`. They are responsible for interpreting the `queryString` inside the JRXML and producing a `JRDataSource`. Some JasperServer data sources don't need `queryStrings` and will produce a `JRDataSource` which is passed directly to the filler with no query executor involved. However, the ability to pass a `queryString` gives the data source a lot more flexibility. In this case, the data source will put implementation-specific objects in the report parameter map that get passed down to the `JRQueryExecutor`, which uses them to produce the `JRDataSource`. The prime example of this is the JDBC data source, which puts a `JDBC Connection` object in the parameter map. The `JRJdbcQueryExecutor` then uses the connection to create a `JDBC Statement` using the `queryString`, run it on the connection, and generate a `JRDataSource` with the result set.

The webscraper example below shows examples of both approaches—it can be used either with or without a `queryString` in the JRXML.

## 6.1.2 Custom Data Source Examples

The examples are found in <js-install>/samples/customDataSource. Once you have deployed a JasperServer web application to your application server, you can use Ant to build and deploy the examples.

If you used an installer to install JasperServer version 2.1 or later, you will have Ant installed already. Ant may be run with the following command:

```
<js-install>/ant/bin/ant <ant-arguments>(for Linux/Unix)
```

```
<js-install>\ant\bin\ant <ant-arguments> (for Windows)
```

If you installed JasperServer manually with a WAR file, you will need to download Ant from <http://ant.apache.org>. Ant 1.6.2 was used for testing, but earlier versions should also work.

The JVM used for installing the examples needs to be a full Java Development Kit, because it needs to compile Java source files. Ensure that the `JAVA_HOME` environment variable points to a JDK installation.

The sample directory includes the following files and directories:

build.xml: The Ant build file

src: Java source directory

webapp: A directory containing other files required by the examples, such as JSPs and Spring configuration

files, which are copied directly to the JasperServer web application directory

reports: A directory containing example JRXML files that use the sample custom data sources

Take the following steps to install the samples in your JasperServer web application (this can be built from the source code or the delivered version of JasperServer):

At the command line, change directories to the custom data source sample directory (`<js-install>/samples/customDataSource`)

Edit `build.xml` and set the `webAppDir` property to the root of the JasperServer web application.

Run the Ant command (see above) with no arguments, which will execute the default target, which is named `deploy`. The `deploy` target will run the following tasks:

Compile the Java source under the `src` directory

Deploy the compiled Java class files to the JasperServer web application

Deploy files under the `webapp` directory to the the JasperServer web application

Restart the application server

### **6.1.2.1 Custom Bean Data Source**

The custom bean data source implementation creates a data source from a collection of Java beans declared in the source code. Its Spring bean definition file is located in `<js-install>/samples/customDataSource/webapp/WEB-INF/applicationContext-sampleCDS.xml`. It An example of a report using this data source is located in `<js-install>/samples/customDataSource/reports/simpleCDS.jrxml`.

### **6.1.2.2 Webscraper Custom Data Source**

The webscraper custom data source implementation can fetch a web page, decode the HTML, and extract selected data which is turned into field values in the data source. Its Spring bean definition file is located in `<js-install>/samples/customDataSource/webapp/WEB-INF/applicationContext-webscraperDS.xml`.

These are the configuration items for the datasource:

URL: An HTTP URL which refers to the HTML page containing the desired content

DOM path: An XPath expression which locates HTML elements to be turned into rows in the data source

Field paths: XPath expressions for each field defined in the JRXML which are used to locate the field value within each row selected by the DOM path

The implementation creates a data source by taking the following steps:

Uses the URL to issue a GET request for an HTML page.

Converts the HTML response into XML using JTidy (<http://jtidy.sourceforge.net>).

Uses the DOM path to select XML elements from the converted response.

Create a new data source row for each selected element

For each field, use the field path to determine the content for each field

The data source has two parameters: the URL of the web page, and the XPath which determines that elements in the HTML page become rows in the data source. The parameters can be specified either by a data source definition in the repository or by a query string in the JRXML.

The example reports for this data source read a web page from <http://www.craigslist.org> and extract a list of items for sale. The report `reports/webscrapertest.jrxml` has no query defined. Instead, it relies on an instance of the custom data source that has been created in the repository. Typical parameters to use with this data source are:



URL: <http://sfbay.craigslist.org/search/car/eby?query=&neighborhood=62>

DOM Path: /html/body/blockquote[2]/p

The `reports/webscraperQEtest.jrxml` example contains a `queryString` element which specifies the URL and the DOM path. It should be used without defining a data source instance, because JasperServer will not run the query executor for this particular implementation if a data source is defined for the report unit.

### 6.1.3 Creating a Custom Data Source

A custom data source consists of Java code, a message catalog, and a Spring bean definition file that configures all the parts of the implementation with JasperServer. This section defines the steps for implementing a custom data source.

#### 6.1.3.1 Files used by a custom data source implementation

Type of files	Path (relative to web application directory)	Description
Spring bean definition	WEB-INF/applicationContext-name.xml	Defines Spring beans needed to configure the data source. Choose a unique name starting with "applicationContext-" and ending with ".xml"
Message catalog	WEB-INF/bundles/xyz.properties	Defines messages used by the data source implementation (this path is referenced in the Spring bean definition file).
Implementation classes	WEB-INF/lib or WEB-INF/classes	Any Java code required by the implementation.

#### 6.1.3.2 Implementing the `ReportDataSourceService` Interface

A custom data source requires an implementation of the `ReportDataSourceService` interface, which is responsible for setting up and tearing down data source connections in JasperServer. It relies on:

`void setReportParameterValues(Map parameterValues):` called before running a report - creates resources needed by JasperReports to obtain a `JRDataSource` and adds them to the parameter map

`void closeConnection():` clean up any resources allocated in `setReportParameterValues()`

#### 6.1.3.3 Defining Custom Data Source Properties

A custom data source can define properties that can be used to configure each data source instance differently, in the same way that a JDBC datasource has properties for JDBC driver class, URL, user name, and password. The definition of properties is covered later (refer to section Error: Reference source not found "Error: Reference source not found" on page Error: Reference source not found), but you will need to consider what properties you want to use while implementing your `ReportDataSourceService`.

There are two kinds of properties:

- Editable properties are string values. When you use the JasperServer data source wizard to create an instance of your custom data source, you can enter values for the editable properties using a text field. These values are persisted when you save the data source.
- Hidden properties can be of any type, but their values are determined by the Spring configuration file, so they are not persisted, and are not visible in the data source wizard. They can be used to give your `ReportDataSourceService` implementation access to a Spring bean instance. For an example of a hidden property, see the `repository` property in the custom bean data source definition below.

These property values are set by the custom data source framework after it instantiates your `ReportDataSourceService` implementation. You need property setters and getters corresponding to the property name; for example, if you defined a property with the name `foo` you will need `getFoo()` and `setFoo()` methods.

#### 6.1.3.4 Implementing Optional Interfaces

If you wish to use the value of the `queryString` in the JRXML to obtain your data source, you need to create implementations of the `JRQueryExecuter` and `JRQueryExecuterFactory` interfaces.

`JRQueryExecuterFactory` has this method:

- `JRQueryExecuter createQueryExecuter(JRDataset dataset, Map parameters):` return a `JRQueryExecuter` for the given dataset and parameter map.
- `JRQueryExecuter` has these methods:
- `JRDataSource createDatasource():` returns the actual data source based on the parameter map that was passed to the `JRQueryExecuterFactory` above; most likely, you will create a `JRDataSource` implementation suitable for your data source.
- `close():` called when report filling process is done with the data source.
- `cancelQuery():` called to clean up resources if the report filling process is interrupted.
- If you wish to provide validation in the user interface for creating and editing custom data source instances, you need to create an implementation of `CustomDataSourceValidator`. It has the following method:
- `validatePropertyValues(CustomReportDataSource ds, Errors errors):` check parameters and call `errors.rejectValue()` with the appropriate property name and error code (defined in a message catalog, refer to section 6.1.3.5 "Creating the Message Catalog," next).

### 6.1.3.5 Creating the Message Catalog

The message catalog contains messages displayed by the data source wizard when creating and editing custom data source instances. The various types of messages are shown in the table below, along with conventions for choosing message names:

Message Type	Naming Convention
Name of custom data source type	<code>Cdsname.name</code> (where <code>cdsname</code> is the value of the name property of the custom data source).
Name of custom data source property	<code>Cdsname.properties.propname</code> (where <code>propname</code> is the value of the property name).
Validation messages	<code>Cdsname.any.message.code</code> (there is no convention enforced, but JasperSoft recommends starting with <code>cdsname</code> for consistency).

For example, the `webscraper` message catalog contains the following:

```
webScrapDataSource.name=Web Scraper Data Source
webScrapDataSource.properties.url=URL
webScrapDataSource.properties.path=DOM Path
webScrapDataSource.url.required=A value is required for the URL
webScrapDataSource.path.required=A value is required for the DOM path
```

### 6.1.3.6 Defining the Custom Data Source in Spring

To configure the data source, you must add an instance of `CustomDataSourceDefinition` to the Spring bean definition file. This class has the following properties:

Name	Required	Value
<code>factory</code>	▲	(Fixed value) <code>ref="customDataSourceFactory"</code> (this is the bean that manages all the custom data sources).
<code>name</code>	▲	Unique name used to refer to custom data source in messages, etc.
<code>serviceClassName</code>	▲	A class name for your <code>ReportDataSourceService</code> implementation.
<code>validator</code>		An instance of your <code>CustomDataSourceValidator</code> implementation.

propertyDefinitions		A list containing a map of information about each property used by the custom data source (details on map entries are listed in the table below).
queryExecuterMap		Map with query languages (uses language attribute of JRXML queryString element) as keys, and JRQueryExecuterFactory class names as values.

The `propertyDefinitions` property is a list of maps, each one describing a property of the custom data source implementation. These are the entry keys used currently:

Name	Required	Value
name	▲	Name of property, which matches a JavaBean property in the <code>ReportDataSourceService</code> implementation, and is also used in message catalog keys
default		A default value for the property
hidden		Properties with hidden set to true are set to fixed values using the "default" entry. They are not be editable in the UI or persisted. This is handy for making Spring beans accessible to <code>ReportDataSourceService</code> implementations

The following XML creates a `CustomDataSourceDefinition` bean for the custom bean data source example:

```
<bean id="myCustomDataSource"
class="com.jaspersoft.jasperserver.api.engine.jasperreports.util.CustomDataSourceDefinition">
  <property name="factory" ref="customDataSourceServiceFactory"/>
  <property name="name" value="myCustomDataSource"/>
  <property name="serviceName" value="example.cds.CustomSimplifiedDataSourceService"/>
  <property name="validator">
    <bean class="example.cds.CustomTestValidator"/>
  </property>
  <property name="propertyDefinitions">
    <list>
      <map>
        <entry key="name" value="foo"/>
      </map>
      <map>
        <entry key="name" value="bar"/>
        <entry key="default" value="b"/>
      </map>
      <map>
        <entry key="name" value="repository"/>
        <entry key="hidden" value="true"/>
        <entry key="default" value-ref="repositoryService"/>
      </map>
    </list>
  </property>
</bean>
```

### 6.1.3.7 Configuring the Message Catalog

To configure your message catalog with JasperServer, add a bean definition similar to the following to the Spring definition file that you created above. For the value of the `value` property, substitute the location of your message catalog file, omitting the `.properties` extension. It is not necessary to edit the `messageSource` bean definition in `applicationContext.xml`.

```
<bean class="com.jaspersoft.jasperserver.api.common.util.spring.GenericBeanUpdater">
  <property name="definition" ref="addMessageCatalog"/>
  <property name="value" value="WEB-INF/bundles/cdstest"/>
</bean>
```

## 6.1.4 Installing a Custom Data Source

Add all the files required by the custom data source implementation to the JasperServer web application and restart the application server.

The new custom data source type appears in the list of choices for data source type when you create a new data source in JasperServer. If the new type is selected, JasperServer displays a form containing the list of properties you configured.

When the form is submitted, the parameter values are validated with the `CustomDataSourceValidator` implementation and appropriate validation messages are displayed.

### 6.1.5 Using a Custom Data Source

When defining `<queryString>` in JRXML, use a `language` setting that your custom data source supports.

When you create a report and are prompted for the data source to use, any custom data sources created by the administrator are displayed.

If you select a locally defined data source, you can choose the new custom data source type and edit it, just as if you were creating a data source in a folder (see section 6.1.4 “Installing a Custom Data Source” on page 63).

## 6.2 Report output channels

When running a report in JasperServer, the result can be delivered in several supported formats such as HTML, PDF, RTF, XLS and CSV. The report viewer page, where reports are viewed initially in HTML format, has a toolbar on top with buttons for exporting the current report to the above mentioned formats. A similar document export process occurs when reports are scheduled and the output is delivered in one of these formats.

The document export process is controlled using the `WEB-INF/applicationContext.xml` file where for each of the supported export formats there is a bean with properties that help fine-tune the export.

### 6.2.1 Excel output

The Excel output channel is controlled by the `xlsExportParameters` bean inside the `WEB-INF/applicationContext.xml` file. JasperServer is shipped with a default configuration that produces a data-centric Excel output where graphic elements are ignored, spacer cells are removed and the data type of cells is preserved.

The following properties can be set for the Excel exporter:

Property Name	Default	Description
<code>detectCellType</code>	<code>true</code>	Preserve the type of the original text field expressions and use it for the cell data type
<code>onePagePerSheet</code>	<code>false</code>	Each report page should be written in a different XLS sheet
<code>removeEmptySpaceBetweenRows</code>	<code>true</code>	Empty spaces that could appear between rows should be removed or not.
<code>whitePageBackground</code>	<code>false</code>	Force cell white background
<code>ignoreGraphics</code>	<code>true</code>	Export text elements only
<code>collapseRowSpan</code>	<code>true</code>	Collapse row span and avoid merging cells across rows
<code>ignoreCellBorder</code>	<code>true</code>	Do not draw the cell border
<code>fontSizeFixEnabled</code>	<code>true</code>	Decrease font size so that texts fit into the specified cell height
<code>maximumRowsPerSheet</code>	<code>0</code>	Maximum number of rows allowed before continuing on a new sheet

In report templates, Java specific format patterns are used to format dates and numbers (see JasperReports documentation for details). But these patterns might not work inside the Excel document viewer program, as their syntax might not be fully supported there.

In such cases, the Java format patterns have to be converted to equivalent proprietary format patterns using the

---

`formatPatternsMap` bean inside the `WEB-INF/applicationContext.xml` file, where pairs of equivalent patterns can be added.

## 6.2.2 CSV output

The CSV export options can be changed by modifying the `csvExportParameters` bean configuration inside the `WEB-INF/applicationContext.xml` file.

For the moment, only the field delimiter can be configured for the CSV output channel.

## 6.3 Configuring Login Page

Starting with version 2.1, JasperServer comes with a new login page which displays welcome information about the product. If you want to switch to the simpler login page that was shipped prior to version 2.1, you can alter the `paramResolver` bean inside the `WEB-INF/applicationContext.xml` file by changing the line:

```
<prop key="/login.html">login_welcome</prop>
```

into

```
<prop key="/login.html">login</prop>
```

## 7 Integrating JasperServer and Liferay Portal

These instructions assume that both JasperServer and Liferay have been installed properly. For instructions about installing Liferay, refer to its documentation. For instructions about installing JasperServer, refer to the *JasperServer Professional Installation Guide*. This chapter also assumes:

- You are using Liferay 5.0 Final Release, bundled with Tomcat 5.5 for JDK5.0, which can be downloaded from:

<http://sourceforge.net/projects/lportal/>

- `<js-install>` is the root of your JasperServer installation.
- `<liferay-install>` is the root of your Liferay installation. This path cannot contain any spaces.

These instructions describe how to deploy the portlet WAR that is included in the JasperServer Professional distribution. If you are implementing the JasperServer portlet from a different distribution, refer to the instructions provided in that distribution.

If you are upgrading from a previous version of JasperServer in which you deployed the Liferay portal, you must take additional steps; pay careful attention when following the instructions in this chapter.

### 7.1 Changing Liferay's Port Numbers

By default, both Liferay and JasperServer run on port 8080. You must change Liferay's listening port to avoid this conflict. For example, you can change Liferay's listening port to 7080. You must also update a number of other Liferay ports.

To configure Liferay's port numbers:

1. In the `server.xml` file (found at `<liferay-install>\conf\`), change all the occurrences of port numbers. For example:

Port	Default	Suggested
Server Port	8005	7005
Non-SSL Connector Port	8080	7080
AGP 1.3 Connector Port	8009	7009
Proxy Connector Port	8082	7082

- In the `server-minimal.xml` file (found at `<liferay-install>\conf\`), change all the occurrences of port numbers. For example:

Port	Default Value	Suggested Value
Server Port	8005	7005
Catalina Connector Port	8080	7080
AGP 1.3 Connector Port	8009	7009

- Start the Liferay Portal by entering the following at the command prompt:

Linux	<code>&lt;liferay-install&gt;/bin/startup.sh</code>
Windows	<code>&lt;liferay-install&gt;\bin\startup.bat</code>

## 7.2 Configuring JasperServer to Accept Web Services Calls

Configure JasperServer to accept web service calls from Liferay by updating

`<js-install>/webapps/jasperserver-pro/WEB-INF/applicationContext-security.xml`.

To configure JasperServer to accept web services from trusted hosts:

- Start JasperServer.
- Locate the `trustedIpAddress` property in the `applicationContext-security.xml` file found at `<js-install>/webapps/jasperserver-pro/WEB-INF`.
- Add the IP address of your Liferay Portal host. For example, if Liferay is running on a computer with the IP address `172.17.3.171`, the `trustedIpAddress` property would be:

```
<property name="trustedIpAddress">
  <list>
    <value>172.17.3.171</value>
  </list>
</property>
```

Jaspersoft strongly recommends using a static IP for this host. Do not use the value `localhost` or `127.0.0.1` for this property.

- Restart JasperServer.

## 7.3 Configuring Liferay to Access JasperServer

The JasperServer portlet uses web services to retrieve information from JasperServer. This parameter specifies the JasperServer repository web service URL. It's normally of the form:

`http://<host:port>/jasperserver-pro/services/repository`

or

`https://<host:port>/jasperserver-pro/services/repository`

For example:

`http://172.17.3.171:7080/jasperserver-pro/services/repository`

To configure Liferay to access JasperServer:

1. Locate the `jasperserver_repository_ws_url` parameter of the `portlet.xml` file found at `<liferay-install>/webapps/<context_name>/WEB-INF`.
2. Set its value to the URL of the JasperServer repository web service. For example:

```
<init-param>
  <name>jasperserver_repository_ws_url</name>
  <value>
    http://172.17.3.171:8080/jasperserver-pro/services/repository
  </value>
</init-param>
```

3. Restart Liferay.

## 7.4 Testing Liferay

To test Liferay:

1. Enter your Liferay Portal URL in your browser's address bar. For example:

`http://<liferay host>:7080/c/portal`

2. Login to Liferay; by default Liferay includes a default Admin user with these credentials:
  - User name: `test@liferay.com`
  - Password: `test`

If the portal appears, Liferay is properly installed.

## 7.5 Deploying the JasperServer Portlet WAR File

If you are upgrading from a previous version of JasperServer in which you deployed the portlet WAR file, you must delete the `webapps/Jaspersoft` folder of the application server hosting Liferay. This deletes libraries used in older versions that conflict with libraries in the latest version. Once this folder is deleted, you can deploy the new portlet WAR.

To deploy the portlet WAR file:

1. Login on the Liferay Portal as a user with administrative privileges. For example, `test@liferay.com`. The default password is `test`.
2. Add the Admin portlet to your current dashboard by clicking **Add Content > Admin > Admin portlet**.
3. Click the **Plugins** tab in the Admin portlet.
4. Click **Install More Portlets**.

5. Click the **Upload File** tab.
6. Locate and add the JasperServerPortlet WAR file
7. Select an optional context.
8. Click **Install**.

Liferay Portal attempts to deploy the WAR file (this step may take several moments, depending on your environment).

9. When Liferay finishes the deployment, click **Add Content > Jaspersoft**. The **JasperServer Portlet** appears.

The JasperServer portlet is deployed.  
Note that you can run multiple instances of the portlet at once.

## 7.6 *Configuring a Default Report to Display*

By default, the portlet displays all the reports that the current user is allowed to view. You can configure the portlet to display a specific report, instead.

In the portlet.xml file, add the following entries to specify a default report:

- `full_resource_path`: This parameter specifies the full report Path.
- `resource_type`: The only supported value is report
- `number_of_parameters`: This specifies the number of parameters the report requires, including optional and required parameters.
- `js_resource_parameter_<parameter>`: For each parameter of the report, append prefix `js_resource_parameter_` to the name. In this example, the report `/reports/samples/EmployeeAccounts` takes a parameter named `EmployeeID`; it is defined as `js_resource_parameter_EmployeeID`.
- `modifiable`: This value has to be 1 [one] so end users can choose different reports at run time. 1 is the only supported value. For details of this parameter, please refer to the JSR-168 specification.

For example, the portlet configuration section of the `portlet.xml` file might be similar to the following:

```
<portlet-preferences>
  <preference>
    <name>full_resource_path</name>
    <value>/reports/samples/EmployeeAccounts</value>
    <modifiable>1</modifiable>
  </preference>

  <preference>
    <name>resource_type</name>
    <value>report</value>
    <modifiable>1</modifiable>
  </preference>

  <preference>
    <name>number_of_parameters</name>
    <value>1</value>
    <modifiable>1</modifiable>
  </preference>
</portlet-preferences>
```



```

    <preference>
      <name>js_resource_parameter_EmployeeID</name>
      <value>beth_id</value>
      <modifiable>1</modifiable>
    </preference>
  </portlet-preferences>

```

If you remove this section, the JasperServer Portlet displays a list of reports that the current user can access.

## 7.7 Testing the JasperServer Portlet

You can configure several parameters that control the JasperServer portlet. They are located in the `<liferay-install>/webapps/<context_name>/WEB-INF/portlet.xml` file. You must restart Liferay for these changes to take effect.

Note: The values described in this section are case sensitive.

## 7.8 JasperServer Portlet Configuration Options

You can configure several parameters that control the JasperServer portlet. They are located in the `<liferay-install>/webapps/<context_name>/WEB-INF/portlet.xml` file. You must restart Liferay for these changes to take effect.

Note: The values described in this section are case sensitive.

### 7.8.1 Portlet Parameters

#### 7.8.1.1 *image\_servlet\_ssl\_enabled*

An image servlet is used to serve images including pictures and charts. If the servlet is configured to use regular HTTP, this value should be false. If HTTPS is used, the value should be set to true. For example:

```

<init-param>
  <name>image_servlet_ssl_enabled</name>
  <value>>false</value>
</init-param>

```

#### 7.8.1.2 *portal\_server*

The only supported value for this parameter is liferay. For example:

```

<init-param>
  <name>portal_server</name>
  <value>liferay</value>
</init-param>

```

#### 7.8.1.3 *show\_logo*

This parameter indicates whether the company logo (at the top right-hand corner of the portlet) is displayed. Any value other than true hides the logo. For example:

```

<init-param>
  <name>show_logo</name>
  <value>>true</value>
</init-param>

```

#### 7.8.1.4 *show\_return\_to\_report\_list\_icon*

The parameter indicates whether the **Return to Report List** icon is displayed. Any value other than true hides

the icon. For example:

```
<init-param>
  <name>show_return_to_report_list_icon</name>
  <value>true</value>
</init-param>
```

#### 7.8.1.5 *show\_return\_to\_parameter\_icon*

The parameter indicates whether the **Report Options** icon should be shown. Any value other than true hides the icon. For example:

```
<init-param>
  <name>show_return_to_parameter_icon</name>
  <value>true</value>
</init-param>
```

#### 7.8.1.6 *company\_logo*

This parameter specifies the image to use when show\_logo is set to true. The picture must be located under `<liferay-install>/webapps/<portlet_context_name>/WEB-INF/images/`. For example:

```
<init-param>
  <name>company_logo</name>
  <value>jaspersoft-logo.png</value>
</init-param>
```

#### 7.8.1.7 *company\_logo\_hyper\_link*

This parameter specifies the hyperlink URL to request when a user clicks the company logo. For example:

```
<init-param>
  <name>company_logo_hyper_link</name>
  <value>http://www.jaspersoft.com</value>
</init-param>
```

#### 7.8.1.8 *report\_directory*

This parameter specifies a folder in the JasperServer repository that should be used to populate the Report List page in the portlet. For example:

```
<init-param>
  <name>report_directory</name>
  <value>/</value>
</init-param>
```

#### 7.8.1.9 *number\_of\_reports\_per\_page*

This parameter specifies the pagination limit for the Report List page. The value specifies how many reports to display on each page. When there are more reports than the number specified for this parameter, end users must click a link to go to next page. For example:

```
<init-param>
  <name>number_of_reports_per_page</name>
  <value>5</value>
</init-param>
```

## 7.8.2 Example Server and Browser Configuration

This section shows examples of the updated files that must be edited to enable Liferay integration with JasperServer.

Note: Do not specify localhost or 127.0.0.1 for any of the following settings. Otherwise, pictures and charts may

not be displayed.

### 7.8.2.1 *applicationContext-security*

**Default Location:** <js-install>/webapps/jasperserver-pro/WEB-INF/

```
<property name="trustedIpAddress">
  <list>
    <value>172.17.3.171</value>
  </list>
</property>
```

### 7.8.2.2 *portlet.xml*

**Default Location:** <liferay-install>/webapps/<context\_name>/WEB-INF/

```
<init-param>
  <name>jasperserver_repository_ws_url</name>
  <value>
    http://172.17.3.171:8080/jasperserver-pro/services/repository
  </value>
</init-param>
```

### 7.8.2.3 *Browser URL*

Users point their browsers to: [http://Liferay\\_host:7080/c/portal](http://Liferay_host:7080/c/portal)

## 7.9 *Setting up JasperReports Hyperlinks for Use in a Portlet Environment*

Because the portal environment is different than that of JasperServer when runs stand-alone, the behavior of links in reports is also different. Although any valid JasperReports link is rendered within the portal environment, the hyperlinks behave differently than if the report is run directly in JasperServer. For example, some hyperlinks might become static text when running within a portal server.

The JasperServer portlet supports these types of links:

Hyperlink	Reference Tab	Link Parameters Tab	ToolTip Tab
Points to an external URL when the Hyperlink Type is CUSTOM or Reference	Hyperlink Reference Expression value must start with http:// or https://.  When users click the link within a portlet, the current browser window displays the new web page. The user leaves the portal environment.  If Hyperlink is invalid, static text is rendered without links.	Parameters and values that the target web site receives.	ToolTip for the hyperlink.
Points to a particular page of the same report when the Hyperlink Type is LocalPage.	Hyperlink Reference Expression value specifies the target page number of the current report.	Parameters and values that the target web site receives. For multiple-valued parameters, use the same parameter name for each parameter value.	ToolTip for the hyperlink.

Points to a particular page of a different report when the Hyperlink Type is RemotePage.	Hyperlink Reference Expression is used in specification of the target page URI.	Hyperlink Page Expression value specifies the target page number of the target report.	Tooltip for the hyperlink.
--	---	--	----------------------------

Notes:

- This section assumes you are using iReport 3.1.0 to set up the hyperlinks.
- If type `None` is chosen, no hyperlink is rendered.
- For both Hyperlink Target values `Self` and `Blank`, the target report is displayed in the same portlet window.
- For Hyperlink Type `RemoteAnchor` and `LocalAnchor`, no links are rendered.